Dissertation

AGENT-BASED SIMULATION OF HUMAN WAYFINDING: A PERCEPTUAL MODEL FOR UNFAMILIAR BUILDINGS

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines
Doktors der technischen Wissenschaften unter der Leitung von

O.Univ.Prof. Dipl.-Ing. Dr.techn. Andreas Frank
E127
Institut für Geoinformation und Landesvermessung

eingereicht an der Technischen Universität Wien
Fakultät für Technische Naturwissenschaften und Informatik

von

Dipl.-Ing. Martin Raubal, M.S.
Matrikelnummer 8607502
Endresstrasse 11/11
1230 Wien

Wien, Oktober 2001

Dissertation

AGENT-BASED SIMULATION OF HUMAN WAYFINDING: A PERCEPTUAL MODEL FOR UNFAMILIAR BUILDINGS

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Technical Sciences

submitted to the Vienna University of Technology

Faculty of Science and Informatics

by

Dipl.-Ing. Martin Raubal, M.S.

Endresstrasse 11/11

1230 Wien

Advisory Committee:

Univ.Prof. Dipl.-Ing. Dr.techn. Andrew U. Frank

Institute for Geoinformation

Vienna University of Technology

Univ.Prof. Dipl.-Ing. Dr.techn. Werner Kuhn

Institute for Geoinformatics

University of Muenster, Germany

Vienna, October 2001

For Gwen and Ian,

and my parents.

I dream of rain
I dream of gardens in the desert sand
I wake in pain
I dream of love as time runs through my hand

This desert rose
Each of her veils, a secret promise
This desert flower
No sweet perfume ever tortured me more than this

And as she turns
This way she moves in the logic of all my dreams
This fire burns
I realise that nothing's as it seems

I dream of rain
I lift my gaze to empty skies above
I close my eyes, this rare perfume
Is the sweet intoxication of her love

(from *Desert Rose*, © Sting 1999)

ABSTRACT

Researchers in the areas of human wayfinding, spatial cognition, computer science, and artificial intelligence have developed cognitively based computational models for wayfinding. These models focus primarily on learning a spatial environment and on the exploration of mental representations rather than the information needs for wayfinding. It is important to consider the *information needs* because people trying to find their ways in unfamiliar environments do not have a previously acquired mental representation but depend on external information. The fundamental tenet of this work is that all such information must be presented to the wayfinder at each decision point as *knowledge in the world* .

Simulating people's wayfinding behavior in a cognitively plausible way requires the integration of structures for information perception and cognition in the underlying model. In this thesis we use a *cognizing agent* to simulate people's wayfinding processes in an unfamiliar building. The agent-based model is grounded in the ontology and epistemology of the agent and its environment. Both are derived from human subjects testing using an ecological approach. This leads to two tiers in the conceptual model: simulated states of the environment and simulated beliefs of the agent. The agent is modeled with state, an observation schema, wayfinding strategies, and commonsense knowledge. The wayfinding environment is modeled as a graph, where nodes represent decision points and edges represent lines of movement.

The *perceptual wayfinding model* integrates the agent and its environment within a *Sense-Plan-Act* framework. It focuses on knowledge in the world to explain actions of the agent while performing a wayfinding task. We use the concepts of *affordance* and *information* to describe what kinds of knowledge the agent derives from the world by means of visual perception. Affordances are possibilities for action with reference to the agent. Information such as from signs is necessary for the agent to decide which affordances to utilize. During the navigation process the agent accumulates beliefs about the environment by observing task-relevant affordances and information at decision points. The utilization of a so-called "go-to" affordance, i.e., following a pathway, leads the agent from one node to another where it is again provided with percepts. A successful navigation corresponds to the agent's traversal from a start to a goal node. The perceptual wayfinding

model concentrates on the actual information needs during wayfinding and does not focus on learning a spatial environment.

The proposed formal algebraic specifications of the agent-based model within a functional programming environment can be used to simulate people's wayfinding behavior in spatial information and design systems in a cognitively plausible way. The simulation helps to determine where and why people face wayfinding difficulties and what needs to be done to avoid them. We employ the specific case of wayfinding in an airport to demonstrate the perceptual wayfinding model. The result can be practically used to test the signage in the airport.

KEYWORDS

Human Wayfinding, Agent, Information, Affordances, Spatial Perception and Cognition, Algebraic Specifications, Simulation.

KURZFASSUNG

Forscher in den Bereichen der menschlichen Wegesuche, der räumlichen Kognition, der Informatik und der künstlichen Intelligenz haben in der Vergangenheit zahlreiche auf Kognition basierende Computermodelle für die Wegesuche entwickelt. Diese Modelle versuchen eher das Lernen einer räumlichen Umgebung und ihrer mentalen Repräsentation zu erklären, als die Informationsbedürfnisse während der Wegesuche. Diese *Informationsbedürfnisse* sind aber ein wichtiger Aspekt, denn Menschen, die in einer unbekannten Umgebung versuchen, ihren Weg zu finden, besitzen keine vorher erworbene mentale Repräsentation dieser Umgebung und sind deshalb auf externe Information angewiesen. Die wesentliche Grundannahme dieser Arbeit ist es, dass diese Information an jedem Entscheidungspunkt für den Wegesuchenden als sogenanntes *Wissen in der Welt* gegenwärtig sein muss.

Um menschliches Wegesuchverhalten in einer kognitiv glaubwürdigen Weise simulieren zu können, muss man dafür sorgen, dass Strukturen für die Perzeption und Kognition von Information im zu Grunde liegenden Modell integriert sind. In der vorliegenden Doktorarbeit werden Prozesse der menschlichen Wegesuche in einem unbekannten Gebäude durch einen *kognizierenden Agenten* simuliert. Das Agenten-basierte Modell ist auf die Ontologie und Epistemologie des Agenten und seiner Umgebung gegründet. Sowohl die Ontologie als auch die Epistemologie ist von den Ergebnissen empirischer Tests mit Versuchspersonen mittels einer ökologischen Methode abgeleitet. Dies führt zu zwei Stufen im konzeptuellen Modell, einerseits simulierte Zustände der Umgebung und andererseits simulierte Überzeugungen des Agenten. Der Agent wird dabei durch folgende Komponenten modelliert, nämlich durch seinen gegenwärtigen Zustand, ein Beobachtungsschema, Wegesuchstrategien und Allgemeinwissen. Die Umgebung, in der die Wegesuche stattfindet, wird durch einen Graphen modelliert, wobei die Knoten des Graphen Entscheidungspunkte repräsentieren und die Kanten Bewegungslinien.

In dem hier präsentierten *perzeptuellen Modell der Wegesuche* sind der Agent und seine Umgebung in ein sogenanntes *Sense-Plan-Act* System integriert. Das Modell konzentriert sich auf Wissen in der Welt, um Aktionen des Agenten während der Wegesuche zu erklären. Die Konzepte *Affordanz* und *Information* beschreiben dabei,

welche Wissensarten der Agent aus seiner Umgebung mittels visueller Perzeption aufnimmt. Affordanzen sind Aktionsmöglichkeiten für den Agenten. Information, wie zum Beispiel von Schildern, wird vom Agenten für die Auswahl einer zu nützenden Affordanz benötigt. Während des Navigationsprozesses nimmt der Agent Aufgaben-relevante Affordanzen und Information an Entscheidungspunkten wahr und sammelt somit Überzeugungen über seine Umgebung. Die Ausnützung einer sogenannten „go-to“ Affordanz, das heißt, das Folgen eines Pfades, führt den Agenten von einem Knoten zum nächsten, wo er wieder mit Wahrnehmungen konfrontiert wird. Eine erfolgreiche Wegesuche entspricht der Durchquerung eines Gebäudes von einem Anfangsknoten zu einem Zielknoten. Das perzeptuelle Modell der Wegesuche konzentriert sich auf die gegenwärtigen Informationsbedürfnisse während der Wegesuche, aber nicht auf das Lernen einer räumlichen Umgebung.

Die vorgeschlagenen formalen algebraischen Spezifikationen des Agenten-basierten Modells sind in einer funktionalen Programmierumgebung repräsentiert und können dazu verwendet werden, menschliches Wegesuchverhalten in räumlichen Informations- und Planungssystemen in einer kognitiv glaubwürdigen Weise zu simulieren. Diese Simulation hilft dabei festzustellen, wo und warum Menschen vor Probleme bei der Wegesuche gestellt werden und was getan werden muss, um solche Probleme zu vermeiden. Der spezielle Fall von Wegesuche in einem Flughafen wird in dieser Arbeit zur Demonstration des perzeptuellen Modells der Wegesuche verwendet. Die Resultate können praktisch für einen Eignungstest der Beschilderung im Flughafen verwendet werden.

SCHLÜSSELWÖRTER

Menschliche Wegesuche, Agent, Information, Affordanzen, räumliche Perzeption und Kognition, algebraische Spezifikationen, Simulation.

ACKNOWLEDGMENTS

I want to express my thanks to Andrew Frank, my thesis advisor. It was always a pleasure for me to discuss the different aspects of this work with him. He is a truly interdisciplinary researcher and was therefore able to give advice wherever I needed it. His guidance and ideas were essential for this work. I am also grateful to my second advisor, Werner Kuhn, who provided me with many important comments and suggestions for improvement. I very much enjoyed talking to and thinking with him, whether it was while having a fantastic dinner in Muenster, at the beach in La-Londe-Les Maures, or via e-mail.

I also want to thank my colleagues at the Institute for Geoinformation for discussing my work with me. Special thanks go to my officemate Hartwig Hochmair, also to Stephan Winter, and Steffen Bittner for interesting discussions and corrections; to Gerhard Navratil and Andreas Grünbacher for their formal help; and to Martin Staudinger for his never-ending knowledge of Word.

Various people outside the Institute also helped me with this work. Among them I would like to acknowledge Mike Worboys for working with me on a COSIT paper, Dan Montello for his help on Gibson, and Heinz Horatschek for showing me around the Vienna International Airport.

Last but not least I want to thank my wife Gwen for her love and support. She always had an open ear for me, both in good and in bad times. She encouraged me when I was down and provided me with new motivation when I needed it. On top of all that she corrected the grammar and style of this work, sometimes even at five o'clock in the morning while our little son Ian was still asleep.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

This chapter starts by giving a scientific and pragmatic motivation for the work done in this thesis. The hypothesis states that human wayfinding in an unfamiliar building can be explained on the basis of an agent-based model for perceptual wayfinding. Both our general approach and the specific research design for verifying this hypothesis are explained. Expected results, potential applications, as well as the intended audience are described. Finally, we present the organization of the rest of this thesis.

1.1 Motivation

Wayfinding and orientation form integral parts of people's daily lives. We have to find our ways through cities, through buildings, along streets and highways, using public transportation, etc. Many times the environment to be navigated is unfamiliar: People are there for the first time and have to find a goal without the help of a previously acquired mental map. They depend on external information or what Norman (1988) calls *knowledge in the world*. Such knowledge resides in the environment and is communicated through signs, guidance systems, and architectural clues. In many cases people find it difficult to perform wayfinding tasks in an unfamiliar environment because they are not provided with adequate knowledge in the world. The main reason for environments being too complex to facilitate wayfinding is a deficiency of clues (Raubal and Egenhofer 1998). They either lack sufficient wayfinding information or their architectures are badly designed and therefore not readable. We all know the stressful and sinking feeling when one gets lost in an airport, a large office building, or on a university campus.

The motivation for this thesis is twofold. On the one hand we are concerned with the *scientific question* of how people find their ways in unfamiliar environments. Previous research focused on the development of computational models that simulate wayfinding in familiar environments. Route-planning tasks are solved by using a previously acquired

mental representation of the environment. During wayfinding in an unfamiliar environment people cannot use such representations but have to rely on other sources to satisfy their *information needs* (Gluck 1991). We are interested in how people immediately make sense of the information presented to them at various decision points in the environment and how they proceed further during a wayfinding task using the provided information. This means looking at the wayfinding process itself instead of looking at the representation such as a cognitive map.

On the other hand we are developing a *practical tool*, which allows us to test the wayfinding information provided to people in an environment. Its use to simulate wayfinding tasks—even before actual construction of a built environment—makes it possible to determine where people face wayfinding difficulties, why they face them, and how wayfinding information and design have to be changed to avoid such difficulties. Simulation of human behavior in space is a powerful research method to advance our understanding of the interaction between people and their environment. It allows for both the examination and testing of models and their underlying theory as well as the observation of the system's behavior (Gimblett *et al.* 1997).

The exploration of alternative building designs and wayfinding guidance systems—primarily signs—and the testing of different ideas and theories before implementation in the real world can result in major economic benefits. Badly designed buildings and guidance systems for wayfinding are a potential danger for people during emergency situations and need to be restructured, which is a costly endeavor. For example, during a fire accident wayfinding based on knowledge in the world can decide upon life or death, because people need to find the emergency exits as quickly as possible. Signs showing the way to the next emergency exit are therefore required at every decision point. The economical relevance is also demonstrated by the following case: If passengers in an airport are able to find their gates without having to use maps or seek advice from airport personnel—queuing up at information desks, which costs time—then airlines will save money that they currently lose due to passengers, and therefore airplanes, being late. The departure delay statistics of an Austrian airline company for the year 2000 shows that the

overall time of delay caused by passengers who did not make it to their gate on time was more than 6000 minutes, costing the airline company about 5 Mio. ATS or 360.000 €[1].

1.2 Goal and hypothesis

The goal of this thesis is to develop a computational theory of *perceptual wayfinding*, which explains how people find a specific destination in an unfamiliar building. The theory uses an agent-based approach and focuses on knowledge in the world in the form of affordances and information, and their utilization by the agent during the wayfinding process. People are involved in various activities during wayfinding, therefore the agent needs to include different components whose interplay allow for simulating these activities. More specifically, we want to answer the following two questions:

1. What is the minimum set of components for an agent to find a specific goal in an unfamiliar building?
2. What is the minimum amount of knowledge in the world—affordances and information—necessary for an agent to find a specific goal in an unfamiliar building?

We do not focus on aspects of learning and lasting cognitive-map-like representations of the environment. Our central hypothesis is:

Human wayfinding in unfamiliar buildings can be explained on the basis of an agent-based model for perceptual wayfinding. This model simulates the interaction of a minimum amount of knowledge in the head and knowledge in the world.

The main hypothesis can be further detailed by the following two sub-hypotheses:

1. *The agent needs a minimum set of interacting components—knowledge in the head—to find a specific goal in an unfamiliar building. These components are its observation schema, the agent's state, wayfinding strategies, and commonsense knowledge.*

---

1 If passengers who already checked in for a flight do not arrive at the gate, their luggage has to be removed from the aircraft before take-off due to security reasons. One arrives at the cost by multiplying the delay minutes with the loss of return per delay minute, which in this case is more than 800 ATS or 58 €. These data were acquired from a reliable source through personal communication.

2. *Knowledge in the world can be represented by affordances and information. The process of wayfinding works on the basis of the interplay between the two.*

1.3 Approach and scientific background

We use a particular case study to answer the research questions posed and to verify our hypothesis: A simulated agent has to find a specific gate at the Vienna International Airport. The agent is unfamiliar with this airport environment and depends therefore on knowledge in the world represented by affordances and information on signs. Focusing on this particular domain and task allows us to reduce the complexity of human wayfinding to a manageable level. This does not affect the validity of the results. The practical relevance of the case study was shown in section 1.1.

It is our intention to simulate human behavior in space. We take one possible approach from the field of cognitive psychology for modeling the wayfinding process, i.e., ecological psychology. It deals with the study of the information transactions between living systems and their environments. Our work focuses therefore on properties of the environment as perceived and cognized by humans. Much of the information people need to perform a task is in the world and the human mind is perfectly tailored to make sense of this world (Norman 1988). This principle is essential for navigating an unfamiliar environment and underlies our agent-based wayfinding model, which mainly focuses on *knowledge in the world* and includes only a minimum of *knowledge in the head* the agent needs for successful navigation (Figure 1). This is the reason we call it a model for *perceptual wayfinding*.

The cognizing wayfinding agent is designed based on the *Sense-Plan-Act* framework used in artificial intelligence. Applying this approach leads to a decomposition of the agent system into a sensing system, a planning system, and an execution system. The flow of information is unidirectional and linear. The agent is state-based and has an internal cognitive schema that directs the process of perceiving, deciding, and acting.

Figure 1: *Knowledge in the head* versus *knowledge in the world*.

Formalizing the conceptual model for agent-based wayfinding allows us to describe it more precisely than by using a verbal description and to create a practical tool for simulating our test case. The mathematical and computational approach taken here is the use of executable algebraic specifications in a functional programming language. These specifications are implementation-independent, allow us to check the consistency of the model, and to generate test cases.

1.4 Research design

The research in this thesis is divided into four major parts (Figure 2). We start with the construction of the ontology and epistemology for the agent and its wayfinding environment—a traveler at the Vienna International Airport. Both serve as a foundation for the conceptual agent-based wayfinding model, which is developed during the second part. The third part consists of the formalization of the conceptual model. Finally, we test the formal model by applying it to the case study. The outcome of the simulation is crucial for verifying our hypothesis (section 1.2).

1.4.1 Defining the ontology and epistemology of the agent and its environment

When developing a tool to simulate human behavior in space, one needs to make sure that the underlying theory of the process is firmly grounded in people's real-world experiences. We therefore use empirical data from human subjects testing concerning wayfinding in an airport to construct the ontology and epistemology for the agent and its environment. Both the ontology and epistemology are formed by using an ecological approach, describing on the one hand the content of the airport domain and on the other hand the agent's knowledge of this domain.

1.4.2 Developing the conceptual model for agent-based wayfinding simulation

Previous work in human wayfinding, spatial reasoning and cognition, psychology, philosophy, and existing cognitively based computational models for wayfinding serve as a foundation for the conceptual development of the agent-based wayfinding simulation. We use specific concepts from the fields of artificial intelligence (i.e., agents), ecological psychology (i.e., affordances), and cognitive science (i.e., schema and information) to design the process model. The model is grounded in the ontological and epistemological definition of the agent and its environment (see section 1.4.1) and consists of a cognizing agent that is able to solve goal-based route-finding tasks in an unfamiliar building based on knowledge in the world. We call it a model for *perceptual wayfinding*.

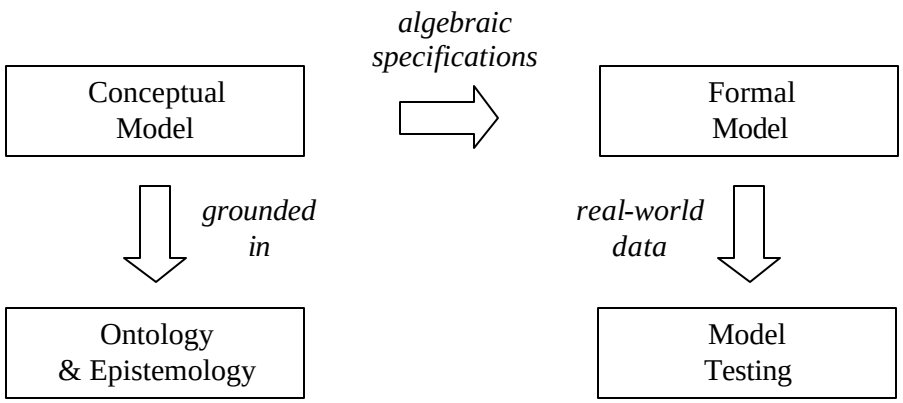

Figure 2: Major research parts of this thesis.

The agent gains knowledge about the building through visual perception of affordances and sign information at decision points. Affordances are what an object, an assemblage of objects, or an environment enables people to do. The agent's perception is directed by an internal observation schema. In this thesis we do not model the process of perception itself, but what kinds of affordances and information the agent needs at each decision point to reach its goal. Neither the ability to learn nor a lasting cognitive-map-like representation of the environment are involved in deciding upon and taking an action. The agent's decisions and actions are founded on wayfinding strategies and commonsense reasoning, and are also guided by the agent's observation schema. Based on knowledge in the world the agent takes a sequence of actions until the wayfinding task is completed. This process is represented through a transition graph consisting of nodes—decision points in the environment—and edges—paths between decision points. A successful navigation of the building corresponds to the agent's traversal of the graph ending at a goal node.

1.4.3 Formalizing the conceptual model

We use algebraic specifications to formalize the conceptual wayfinding model. Algebraic specifications are the link between the conceptual process model and its implementation. The purpose of specifications is to formally describe the behavior of objects and therefore fix the meaning of the conceptual model. For this reason we use them to describe the behavior of the agent performing wayfinding tasks in the airport.

Algebraic specifications of the process model are written in the functional programming language Haskell. Definitions are built in the form of functions, which are evaluated by a computer. Functional programming languages use a similar syntax and have similar mathematical foundations as algebraic specifications; we define data types and operations for these data types. Algebraic specifications written in Haskell are executable and can therefore be tested as a prototype.

1.4.4 Testing the formal model

There are two reasons for testing the formal model. First, we need to clarify and assess the concepts and methods used to develop the agent-based wayfinding simulation. Second, we need to verify our hypothesis. The simulation is applied to our case study: The agent simulates a passenger at the Vienna International Airport and has to perform the task of

finding the way from a check-in counter to a gate. The outcome of the simulation shows whether the agent using the information offered on signs has successfully reached its goal or failed in doing so. Furthermore, we can determine where and why the agent encounters wayfinding problems and what can be done to avoid them. The result of the simulation is a test of the signage in the airport building.

1.5 Major expected results

This work proposes an agent-based process model for perceptual wayfinding to simulate people's information needs for performing wayfinding tasks in unfamiliar buildings. The model focuses on knowledge in the world without paying attention to learning a spatial environment. The major scientific contributions are:

- a formal model for agent-based wayfinding simulation as executable specifications in a functional programming environment and independent of implementation details;
- a conceptual model for agent-based wayfinding simulation that is firmly grounded in people's real-world experiences; it consists of a minimal set of components and integrates elements of human perception and cognition; and
- a method to derive and separate between the ontology and epistemology for a specific domain of human behavior—wayfinding in airports—using an ecological approach; it takes into account the interaction between the agent and its environment and the resulting ontology and epistemology serve as a grounding for the conceptual model;

With wayfinding in an airport as a case study we demonstrate that the computational model can be used as a practical tool to:

- analyze the agent's wayfinding process with regard to success or failure of reaching its goal;
- get details about all perceptions, decisions, and actions of the agent during the wayfinding process;
- determine the positions of decision points responsible for the agent's failure to find its goal;

- find out why the agent took a wrong path at a decision point and what needs to be done to avoid such wayfinding difficulties; and
- test the proposed signage for a building;

The use of the agent-based simulation tool will complement design systems for buildings and should eventually lead to buildings that are easier to navigate and create less stress and anxiety for people. Our model for simulation is defined in such a way that it can be extended and applied to various other wayfinding domains.

1.6 Intended audience

The work done in this thesis is related to several disciplines. It is targeted in particular at researchers in the following areas:

- Architects and designers can use the perceptual wayfinding model to test existing buildings, to test and improve the signage after changes to a building, and to assess possible design alternatives.
- Computer scientists and designers of applications for Geographic Information Systems (GIS) can use the formal specifications to implement the wayfinding model in spatial information and design systems.
- Cognitive scientists and psychologists can apply the perceptual wayfinding model in research on human wayfinding behavior in unfamiliar buildings. The model could be one starting point for human subjects testing in this area.
- Researchers in artificial intelligence can use the formal agent-based model as an important domain of people's everyday lives—wayfinding in an unfamiliar building.

1.7 Organization of the thesis

In the next chapter we present the case study employed in this thesis. It is wayfinding at the Vienna International Airport. Both the specifics of wayfinding in an airport in general and the particular setting of the case study are described. The main task used for the assessment of the agent-based wayfinding model is finding the way from a check-in counter to a gate in the airport. We illustrate this task in detail.

Chapter 3 reviews previous research concerning the modeling of human wayfinding. First, human spatial cognition, which underlies all processes of wayfinding, is introduced. We then describe different aspects of human wayfinding, covering people's abilities and spatial knowledge, reasoning and decision-making, and mental representations. Furthermore, we discuss empirical research, computational wayfinding models, and some relevant concepts from artificial intelligence. All of the presented theories and concepts are linked to the agent-based model for perceptual wayfinding developed in this thesis.

Chapter 4 explains the modeling concepts used in this work. We employ agent as a conceptual paradigm and therefore introduce agent theory including abstract models of agents and their environments. The chapter gives an overview of ecological psychology and Gibson's theory of affordances—including the work done by his critics. The relation between affordances and information for wayfinding is also explained. We propose that wayfinding is based on their interplay. The chapter ends with a description of Neisser's schema theory.

Chapter 5 shows the formal methods used to formally specify the agent-based model for perceptual wayfinding. Graph theory is needed to represent the agent's environment. For the formalization we take an algebraic approach. The concepts behind algebraic specifications and their usefulness are demonstrated. The functional language Haskell provides the programming environment to express the specifications. We describe its main concepts and give examples for its syntax.

In chapter 6 we develop the conceptual model for perceptual wayfinding. We first construct the ontology and epistemology for the agent and the airport environment by employing an ecological approach. They are both based on people's real-world experiences and serve as a grounding for the agent-based model. The two-tiered model is developed according to specific design considerations, which allow us to answer our research questions. It consists of the agent and its environment, and of the perceptual wayfinding process that is represented within the Sense-Plan-Act framework.

Chapter 7 presents the formalization of the conceptual model for perceptual wayfinding. We define classes with operations in Haskell to come up with a computational model consisting of executable algebraic specifications. At the beginning, the data types for the agent and its environment are defined. Then the formal operations of the agent including its main wayfinding strategy are specified. Next, we give the formal

representation of wayfinding information from signs and the agent's additional wayfinding strategy. At the end of the chapter all of the specifications are integrated to form the simulation framework for analyzing the agent's wayfinding process.

In chapter 8 we use the formal model to simulate wayfinding at the Vienna International Airport. This allows us to test the validity of the model and to verify the hypothesis of the thesis. After presenting the test data for the environment and agent we demonstrate three different simulation results for the agent finding its way from a check-in counter to a gate in the airport. The outcomes show whether the agent reaches its goal and if not, where and why the agent faces wayfinding difficulties and what can be done to avoid them. Finally, the assessment of the simulation is presented.

Chapter 9 first summarizes the work done in this thesis. We then present the results and major findings of our research. The chapter concludes with possible directions for future work. The complete Haskell code for the agent-based model for perceptual wayfinding and the complete results for the simulated test cases are given in the Appendix.

CHAPTER 2

THE CASE STUDY – WAYFINDING IN AN AIRPORT

The case study used in this thesis is wayfinding at the Vienna International Airport. We employ it to clarify and assess the concepts and mechanisms underlying the perceptual wayfinding model, and also as a test case for the agent-based simulation. This chapter illustrates the particulars of wayfinding in an airport and the specific setting of our case study. Furthermore, it describes the task of finding one's way from the check-in counter to the gate.

2.1 Wayfinding in an airport

Wayfinding in an airport represents a special case of moving through a building. Passengers in an airport have to find their ways from check-in counters to gates, from gates to the baggage claim area, and between gates. They are often in a hurry and must avoid getting lost. This can be a difficult task, because terminals in general (Arthur and Passini 1992) and airport terminals in particular (Seidel 1982; Brown 2001) are characterized by confusion and disorientation.

Airports do not have a generic shape, they are organized and designed in several ways, and they have a lot of first-time users. Many passengers are unfamiliar with the particular space and fast motion. They come from different cultural backgrounds, speak different languages, and are often late, nervous, and out of focus (Mollerup 2000/01). All of this can put them into stressful situations. Things become even worse in emergency cases such as a fire: During a fire accident at the German Duesseldorf Airport in 1996, sixteen people died because they could not find their way out of the building (Standard 1996). The communication of wrong directions to passengers and the bad architectural design of the airport were blamed for the deadly outcome of this catastrophe.

Airports need clear signage to guide their users. A recent study at the three major airports in New York City has shown a high percentage of passengers getting lost because

of confusing directions. This resulted in the decision to completely redesign the signage system in these airports (Brown 2001). The importance of airport signage is also highlighted by one of the questions asked during the annual airport tests conducted by the International Air Transport Association (IATA): "Is it easy to find my goal without having to ask for directions?"

2.2 The setting of the case study

The Vienna International Airport (VIE) falls into the category of airports serving up to 15 million passengers annually—having served 12 million passengers in the year 2000. A recent survey by the IATA of 90.000 passengers at 48 airports throughout the world ranked it 8^{h} with regard to overall user-friendliness (Flughafen-Wien-AG 2001a). A previous study by the author of this thesis about people's wayfinding difficulties in various airports showed that VIE is considered easy to navigate (Raubal 1997). This result was reinforced by work comparing the complexity of the wayfinding task "going from the departure hall to the gate" at VIE and Frankf urt International Airport in Germany (Raubal and Egenhofer 1998). At VIE the task received a lower rating of points within so-called *problem areas*—decision points with incomplete, misleading, or missing wayfinding clues such as signs and architectural features—indicating a lower wayfinding complexity for passengers.

The airport terminal consists of three levels—the *arrival* level, the *departure* level,

Figure 3: Central part of the departure level at VIE.

Figure 4: Complete map of the departure level at VIE.

and the *restaurant* level. Our case study focuses on the departure level, which comprises two check-in terminals—Terminal 1 and Terminal 2. Figure 3 shows the central part of the departure level indicating the major possibilities for passengers' movements and Figure 4 gives the complete map including the East- and West Pier. The walking distance between these two Piers is approximately 250 meters.

The most important task for departing passengers—and also the main task for our analysis during the agent-based wayfinding simulation—is to find the way from one of the

Figure 5: Check-in counters to the right.

Figure 6: At VIE there exist A-, B-, and C-gates.

check-in counters to their gate. Passengers first have to check in their luggage at one of the check-in counters (Figure 5). They receive a boarding pass, which tells them their boarding gate and the latest time by which they must arrive at this gate. The gates are labeled with the letters *A, B,* or *C* (Figure 6)—denoting the three different gate areas at VIE—and a number. Passengers then have to proceed through the boarding pass and ticket control (Figure 7) to enter the transfer and duty-free area. Before entering the gate, a security control is carried out (Figure 8). Depending on the country of destination—a distinction is made between countries that have signed the Schengen Accord and countries that have not—the passenger may also have to move through passport control (for gates A1 – A19) (Figure 9) or show her passport and boarding pass at the gate desk (for gates C55 – C61) (Figure 10) (Flughafen-Wien-AG 2001b).

Figure 7: Boarding pass and ticket control.

Figure 8: Security control at the gate.

Figure 9: In front of passport control for gates A.

Figure 10: Passport control at the gate.

2.3 Summary

This chapter introduced the case study used in the thesis—wayfinding at the Vienna International Airport. Airports have many particulars and are often characterized by confusion. Together with the unfamiliar environment and stress this can lead to wayfinding problems. Therefore airports should offer clear signs to facilitate wayfinding. The major task for passengers in an airport is finding the way to their gate. We use this task for the analysis with the agent-based wayfinding simulation developed in this thesis.

CHAPTER 3

MODELING HUMAN WAYFINDING

This chapter presents the scientific background on modeling human wayfinding and relates it to the work done in this thesis. Human spatial cognition underlies all processes of human wayfinding and is therefore introduced first. We then review different aspects relevant to human wayfinding: people's abilities and spatial knowledge, spatial reasoning and decision-making, and mental representations. The chapter continues with a discussion of human wayfinding *performance* and its influence on the field of architecture. The final sections of this chapter are devoted to human wayfinding *competence*—i.e., computational wayfinding models—and to the discipline of artificial intelligence and its endeavor to build mobile robots.

3.1 Human spatial cognition

The goal of this thesis is to simulate people's wayfinding behavior in unfamiliar buildings in a cognitively plausible way. We therefore need to represent the relevant aspects of the physical world and integrate elements and concepts of human spatial perception and cognition into the agent-based model. The importance of knowledge about human spatial cognition for explaining and predicting people's behavior in geographic space was stressed by Mark (1997). Geographical cognition is also one of the three strategic research areas of the Varenius project, a U.S. effort to advance geographical information science (Mark *et al.* 1999a).

Human spatial cognition is a part of the interdisciplinary and therefore wide-ranging research area of cognitive science. Researchers from many academic disciplines, such as psychology, linguistics, anthropology, philosophy, and computer science investigate about the mind, reason, experience, and people's conceptualizations of the world in which they live (Lakoff 1987). In particular, cognitive science deals with the study of human intelligence in all of its forms, from perception and action to language and reasoning.

Intelligence can be defined as rational and humanlike thought, i.e., "the ability to attain goals in the face of obstacles by means of decisions based on rational (or truth-obeying) rules" (Pinker 1997, p. 62)[1]. The exercise of intelligence is called cognition (Osherson and Lasnik 1990). The agent specified in this work is able to perceive, decide, and act, and we therefore call it a *cognizing agent*. Cognitive science provides us with the research methods and philosophical positions that allow for a grounding of the agent-based wayfinding model.

Mark *et al.* (1999a) presented a hypothetical information flow model for spatial and geographical cognition, which consists of four stages: acquisition of geographical knowledge, mental representations of geographical knowledge, knowledge use, and communication of geographical information—the first three stages were also given by Neisser (1976) as elements of cognition. This thesis focuses on two of them, that is, modeling the agent's acquisition of affordances and information from its environment and using such knowledge in the world to accomplish a wayfinding task. Furthermore, another stage—acting in the environment—is added.

The term *knowledge in the world* was introduced by Norman (1988) and refers to external information. He argued that much of the knowledge people need to operate things and do certain tasks lies in the world and therefore we are not required to learn everything and store it as internal knowledge or *knowledge in the head* for later use. Norman's main argument is that putting the required knowledge in the world has the effect of reducing people's mental load and leads therefore to easier usability of things. The performance of different tasks requires different tradeoffs between knowledge in the world and knowledge in the head. For wayfinding in unfamiliar environments knowledge in the world is an absolute necessity, because people do not have previously acquired knowledge in the head about the particular environment.

Spatial cognition refers to both the perceptual and conceptual processes that are involved in understanding the physical environment. Therefore, wayfinding theories need to integrate a link between perception and cognition if they want to serve as plausible accounts of people's everyday experience (Allen 1999). We account for this need by including Neisser's (1976) theory of the perceptual cycle in the agent-based process model

[1] The same author states that although it is difficult for us to define intelligence, we recognize it when we perceive it.

for wayfinding. This theory proposes that the perceiver has certain cognitive structures—called *schemata*—that enable the perception and therefore pick-up of information from the environment.

3.2 Human wayfinding

Human wayfinding research investigates the processes that take place when people orient themselves and navigate through space. Theories try to explain how people find their ways in the physical world, what people need to find their ways, how they communicate directions, and how people's verbal and visual abilities influence wayfinding. Lynch (1960, p. 3) defines wayfinding as based on "a consistent use and organization of definite sensory cues from the external environment." Allen (1999) and Golledge (1999) describe wayfinding behavior as purposeful, directed, and motivated movement from an origin to a specific distant destination, which cannot be directly perceived by the traveler. Such behavior involves interactions between the traveler and the environment. The ultimate goal of human wayfinding is to find the way from one place to another. The traveler must be able "to achieve a specific destination within the confines of pertinent spatial or temporal constraints and despite the uncertainties that exist." (Allen 1999, p. 47) Human wayfinding takes place in *large-scale spaces* (Downs and Stea 1977; Kuipers 1978). Such spaces cannot be perceived from a single viewpoint therefore people have to navigate through large-scale spaces to experience them. Examples for large-scale spaces are landscapes, cities, and buildings.

Allen (1999) suggests a *taxonomy of wayfinding tasks* based on functional goals. It consists of three categories:

1. travel with the goal of reaching a familiar destination;
2. exploratory travel with the goal of returning to a familiar point of origin; and
3. travel with the goal of reaching a novel destination.

The work done in this thesis focuses on the task of finding one's way to a novel destination in an unfamiliar environment and therefore exclusively on the third of Allen's categories. In such situations, people have to rely mainly on symbolic spatial information and architectural information communicated to them through the environment, i.e., knowledge in the world. Key processes for this type of communication relate to abilities such as

matching real-world features against knowledge schemas (Raubal 1997) of those features—relating tokens to types or vice versa according to Eco (1999, pp. 179ff.)—and understanding the symbols commonly used to represent real features (Golledge 1999). We do not investigate wayfinding guided by maps or verbal directions in this thesis.

3.2.1 People's wayfinding abilities and spatial knowledge

According to Golledge (1999), human wayfinding refers to people's cognitive and behavioral abilities to find a way from an origin to a destination. These abilities are a necessary prerequisite for people to use environmental information—i.e., knowledge in the world—or representations of spatial knowledge about the environment—i.e., knowledge in the head—to successfully perform wayfinding.

Recent work by Allen (1999) groups people's spatial abilities according to their function, that is, to the tasks and situations in which they are applied. This classification is based on previous research in the psychometric, information-processing, developmental, and neuropsychology traditions. It consists of interactions between

1. a stationary observer and small manipulable objects,
2. an observer and moving objects, and
3. a mobile observer and large stationary objects.

Although there are encounters with moving people and objects, wayfinding in a building is mainly concerned with the third group because people move through an environment that contains large immobile objects. The foundation for this group of spatial abilities is sensitivity to available perceptual information—visual, auditory, vestibular, tactile, or proprioceptive. Examples are obstacle avoidance and path integration.

People's spatial abilities seem to depend mainly on the following four interactive resources: perceptual capabilities, fundamental information-processing capabilities, previously acquired knowledge, and motor capabilities (Allen 1999). These resources support different wayfinding means as can be seen in Figure 11. Such a model could function as a framework for investigating individual wayfinding differences, i.e., why some people are better wayfinders than others. Resource limitations within different groups of people such as handicapped people or people of different age have a definite impact on the success of wayfinding.

As for the spatial abilities, the cognitive abilities also depend on the task at hand. Finding one's way in a street network (Timpf *et al.* 1992; Car 1996) uses a different set of cognitive abilities than navigating from one room to another in a building (Gärling *et al.* 1983; Moeser 1988). People are usually good in applying their individual skills to the task at hand. If their spatial skills are weak, they use verbal skills to navigate—when people get lost, they usually ask someone for help—and vice versa (Vanetti and Allen 1988).

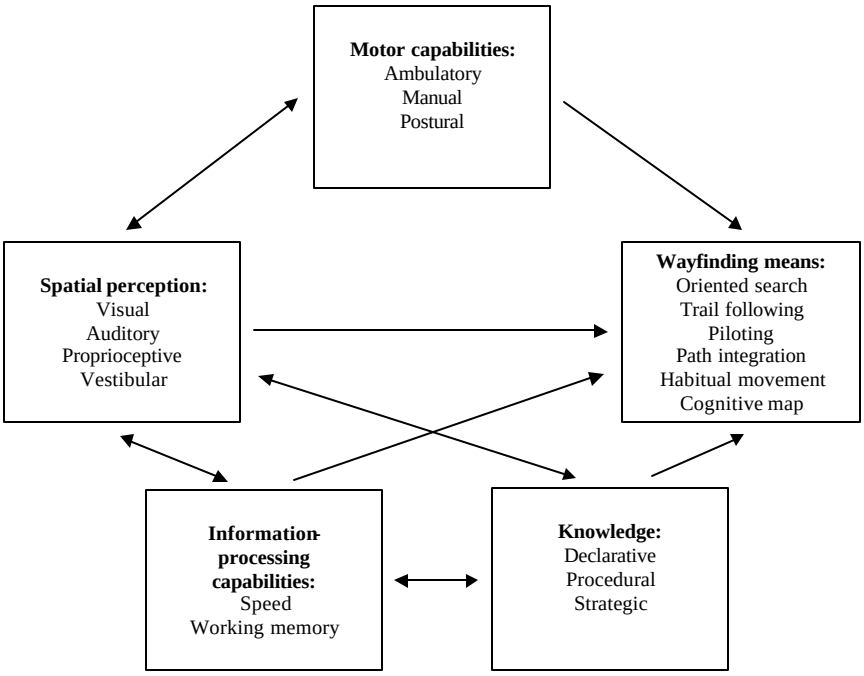

Figure 11: Speculative model of relations among interactive resources and wayfinding means from (Allen 1999, p. 79).

Human spatial knowledge of geographic space is assumed to develop in three successive stages (Siegel and White 1975):

1. *Landmark knowledge* comprises salient points of reference in the environment,
2. *route knowledge* puts landmarks into a sequence (e.g., navigation paths), and
3. *survey* or *configurational knowledge* allows people to locate landmarks and routes within a general frame of reference (i.e., incorporating Euclidean measurements).

This model has been criticized recently for its strict developmental sequence. Montello (1998) proposes a new framework for people's acquisition of spatial knowledge in large-scale environments. He argues that stages with pure landmark or route knowledge do not exist. There are no qualitative shifts from non-metric to metric forms of knowledge because metric knowledge is obtained right from the beginning of the acquisition process and then further accumulated and refined.

3.2.2 Spatial reasoning and decision-making

Instead of doing exact calculations, people apply qualitative methods of spatial reasoning (Frank 1992; Freksa 1992; Cohn 1995; Frank 1996) that rely on magnitudes and relative, instead of absolute, values. When people perceive space through different channels they arrive at various kinds of information, which are usually qualitative in nature. People do not move through the environment using rulers or tape measures. When viewing a scene the result is a retinal image that is of quantitative nature, but people's knowledge about the scene is qualitative (Freksa 1991). Freksa argues that such knowledge is exactly what people need for the process of spatial reasoning and mentions three advantages:

1. expressive power of qualitative constraints based on their interaction—e.g., the concept of transitivity,
2. independence from specific values and scale, and
3. invariance under transformations;

As an example he introduces the *aquarium metaphor* (Freksa 1991). Two observers look at an aquarium and can locate fish by qualitative means and communicate their spatial perceptions, although they have to deal with incomplete, imprecise, and subjective knowledge. They do this by using qualitative knowledge such as knowledge about

positions of some fish relative to other fish, distinguishing features between fish, and a well-defined context.

People use topological instead of metrical information. Topological properties of objects stay invariant under such transformations as translations, rotations, and scalings. By using abstract geometrical analysis Piaget and Inhelder (1967) demonstrated that fundamental spatial concepts are topological, but not Euclidean at all. They showed that children start to conceptualize space by building up and using elementary topological relationships, such as proximity, separation, order, and enclosure.

Spatial reasoning involves a variety of decision-making methods and choice behavior. Decision theory covers a large range of models with different foci on describing how decisions could or should be made and on specifying decisions that are made (Golledge and Stimson 1997). Mathematically, a *decision rule* is a function that assigns a value to each alternative, showing what will happen when a particular strategy is adopted. *Decision-making criteria* are a set of procedural rules that oversee the evaluation of the outcome when decision rules are applied to a task situation. A *strategy* contains decision rules that seek a result from all possible ways of making a relevant decision.

Classical decision-making theories can be classified into the categories of riskless decision-making, risky decision-making, transitivity in decision-making, and game theory and statistical decision functions. Golledge and Stimson (1997) argue that in many cases human decision-making is not strictly optimizing in an economical and mathematical sense—such as proposed by the algorithms of classical decision-making theories—and therefore emphasize *behavioral decision theory*. In this respect they refer to Timmermans' (1991) typology of decision-making according to spatial choice. It includes models accounting for

1. *variety-seeking behavior* such as in recreational choice,
2. *uncomplicated choice among limited alternatives* such as choice of travel mode,
3. *complex choice situations* including preference and attitude,
4. *temporal choice* involving stochastic models, and
5. *simulation of complicated choice outcomes.*

Spatial choice	*Example from case study*
Uncomplicated choice among limited alternatives	Agent's goal is C57. Agent faces a decision point with two possible continuations—one leading to gate area B, the other one to gate area C => Agent decides for C.
Complex choice situations	Agent's goal is C57. Agent faces a decision point with two possible continuations—one leading to gate areas A and C, the other one to gate areas B and C => Agent decides for A and C based on preference, i.e., preferred directions.

Table 1: Decision-making of the cognizing wayfinding agent.

One needs to distinguish between the *choice act*—the outcome of a decision-making process—and a *preference*—an activity within the decision-making process expressing what is desirable.

Decision-making of the cognizing wayfinding agent as specified in this thesis involves uncomplicated choice among limited alternatives and complex choice situations involving a preference (Table 1). Studies have shown that heuristics for choosing the correct way at an intersection are influenced by configurational parameters of the spatial environment and by people's perspectives during navigation (Janzen *et al.* 2000). Golledge (1995) demonstrated with a series of pilot experiments that people use a variety of path selection criteria, such as shortest distance, least time, and fewest turns, in different contexts. People can apply such criteria only when they are either familiar with the environment or have access to a map of the environment. This is not the case when finding one's way in an unfamiliar building, therefore we propose to model the preference as preferred directions within the agent's egocentric reference frame.

3.2.3 Cognitive maps

When people travel with the goal of reaching a familiar destination they use lasting internal representations of spatial knowledge about the wayfinding environment. One useful metaphor suggests that people have a cognitive map in their heads (Kuipers 1982)—a mental representation that corresponds to people's perceptions of the real world. The term *cognitive map* was first used in a paper by Tolman (1948) who claimed that rats in a maze-learning task acquired knowledge of the spatial relation between start and goal. Other metaphors, such as *cognitive collage* (Tversky 1993) or *cognitive atlas* (Hirtle 1998) have

also been proposed. Neisser (1976) uses the term *orienting schema* as a synonym for cognitive map to stress its active and information-seeking structure instead of defining it as a mental image.

Considering the process of acquiring spatial knowledge of an environment, the cognitive map may develop from a mental landmark map to a mental route map and should eventually result in a mental survey map[2]. The last stage is closest to a cartographic map, although it still contains inaccuracies and distortions. People construct and develop their cognitive maps based on the recording of information through perception, natural language, and inferences. Complex environmental structures can lead to slower development of cognitive maps and also to representational inaccuracies.

Researchers from various disciplines have thoroughly investigated the role cognitive maps play in spatial behavior, spatial problem solving, acquisition, and learning (Kitchin 1994). Much less, however, has been found out about how people immediately understand different spatial situations while performing a wayfinding task. Gluck (1991) points out this lack of information by arguing that previous work on wayfinding concentrated on the description of the cognitive map and neglected affective and logistical concerns in most of the cases. As an alternative approach Gluck suggests exploring the *information needs*. He further envisions a typology of wayfinding scenarios and proposes the use of the *sense-making* investigation method: "'Sense-making' is a creative human process of understanding the world at a particular point in time and space limited by our physiological capacities, our present, past and future." (Gluck 1991, p. 129) The idea behind the sense-making method is to look at the wayfinding process itself instead of looking at the representation of the wayfinding environment.

In this thesis we investigate the information needs of people finding their ways in an unfamiliar environment. We therefore focus on knowledge in the world and do not look at lasting representations. Although Golledge (1999) argues that cognitive maps are necessary for human navigation, people have the ability to find their ways in unfamiliar environments—such as a newly experienced airport or hospital—without referring to a previously acquired cognitive map. One could argue though that people build some sort of mental collage at each bifurcation, which is essentially an integration of different views at a decision point. Such representation helps them in making a decision of how to proceed

[2] It may also develop from a mental landmark map to a mental survey map (see section 3.2.1).

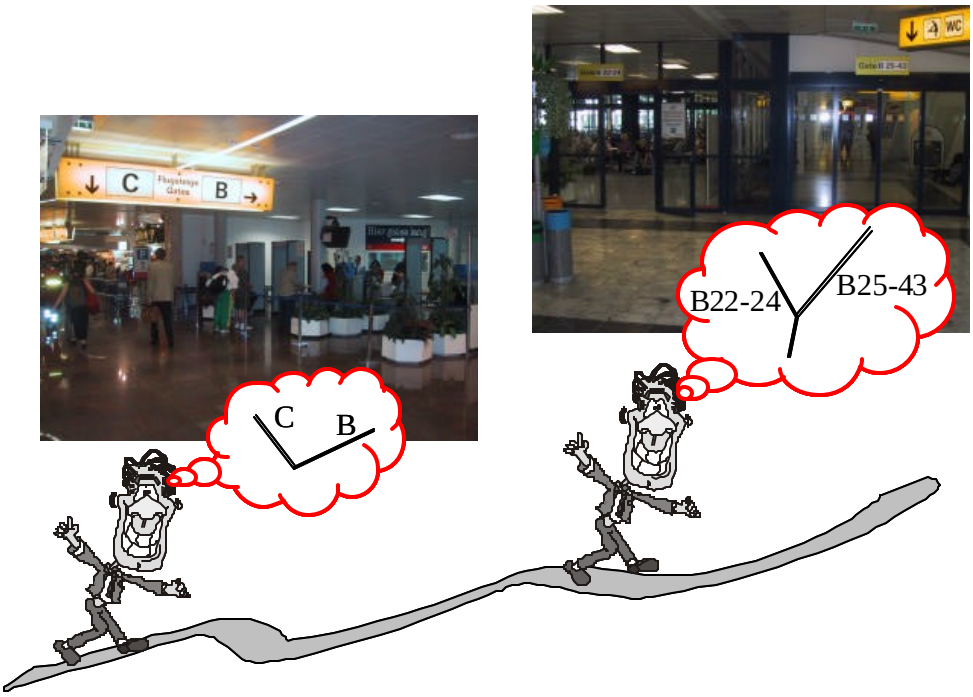

Figure 12: Short-term representations of perceived knowledge at two decision points.

further but the representation does not last long. We specify the simulated agent in this work along these lines: It represents the perceived knowledge in the world for each decision point only at that point in order to apply internal processes including strategic rules to it (Figure 12).

3.2.4 Human wayfinding performance

The literature on *performance* discusses empirical results of how people find their ways. Investigations are based on collecting individuals' perceptions of distances, angles, and locations. An example for a typical experiment is the pairwise judgment of distance between points. Such experiments help in describing features of the cognitive map.

Kevin Lynch's *The Image of the City* (1960) is regarded as the foundation for human wayfinding research. His goal was to develop a method for the evaluation of city form based on the concept of *imageability*—"that quality in a physical object which gives it a high probability of evoking a strong image in any given observer" (Lynch 1960, p. 9)—and to offer principles for city design. Based on his investigations Lynch divided the contents

of the city images into paths, edges (boundaries), regions, nodes, and landmarks. These elements were described as the building blocks in the process of making firm, differentiated structures at the urban scale and have been the basis for later research on wayfinding.

Weisman (1981) identified four classes of environmental variables that influence wayfinding performance within built environments:

1. visual access,
2. the degree of architectural differentiation,
3. the use of signs and room numbers to provide identification or directional information, and
4. plan configuration.

Other researchers confirmed his results. In Gärling *et al.'s* (1983) study of orientation in a large university department visual access was regarded as an important factor, because wayfinding performance—i.e., the accuracy of location estimates—of subjects with restricted sight improved less over time. The impact of orientation tools like floor plans was also investigated. The performance of subjects with restricted sight using floor plans improved as fast as that of subjects without restricted sight, floor plans, therefore, counteracted the negative effect. In another study Gärling *et al.* (1986) proposed to classify the environment by examining the degree of differentiation, the degree of visual access, and the complexity of the spatial layout. The influence of floor plan complexity—e.g., legibility and number of possible paths—on both cognitive mapping and wayfinding performance, and the existence of an interaction between floor plan complexity and the quality of signage were demonstrated in two studies by O'Neill (1991a; b). His results showed that an increase in floor plan complexity leads to a decrease in wayfinding performance—defined as a function of rate of travel in reaching a goal point and the number of wayfinding errors such as a wrong turn. The presence of signage was an important factor but could not compensate for floor plan complexity. Seidel's (1982) study at the Dallas / Fort Worth Airport confirmed that the spatial structure of the physical environment has a strong influence on people's wayfinding behavior. For passengers arriving at the gate with direct visual access to the baggage claim, wayfinding was easier. In addition to Weisman's four classes of environmental variables, people's familiarity with the environment also has a big impact on wayfinding performance: Frequency of prior use

had a big facilitating effect in university buildings (Gärling *et al.* 1983) as well as in airports (Seidel 1982). Cornell *et al.* (1994) tested people's accuracy of place recognition and used the results to develop a model for wayfinding.

Our proposed perceptual wayfinding model is based on people's visual access to affordances and information at decision points. It focuses on signage for directional information and its influence on wayfinding performance. People's familiarity with the environment is not considered as we simulate wayfinding in unfamiliar buildings.

3.2.5 Wayfinding and architecture

Research on people's wayfinding performance has been particularly helpful for establishing practical guidelines on how to design public buildings to facilitate wayfinding. Architects seem to have come to the conclusion that facilitating people's wayfinding needs more than putting up signs, because most of the time signage cannot overcome architectural failures (Arthur and Passini 1992). Therefore, wayfinding principles have to be considered during the design process—both for the overall spatial structure and for the form giving features. Some guidelines (Arthur and Passini 1990; 1992), despite focusing on the design and placement of signage, highly stress the importance of architectural features. In *1-2-3 Evaluation and Design Guide to Wayfinding* Arthur and Passini (1990, p. A-1) introduce the term *environmental communication*—"transfer of orientation, wayfinding (direction), and other information within the built environment by means of signs and other communications devices or architectural features to enable people to reach destinations"—arguing that the built environment and its parts should function as a communication device—see also (Norman 1988).

Arthur and Passini mention two major aspects regarding the understanding of buildings:

1. a *spatial* aspect that refers to the total dimensions of the building—e.g., walls enclose space and elements such as an interior atrium break it up—and
2. a *sequential* aspect that considers a building in terms of its destination routes.

Destination routes should eventually lead to so-called destination zones. These are groupings of similar destinations within buildings into clearly identifiable zones (Arthur and Passini 1992). In order to facilitate wayfinding to such destination zones the

circulation system should be of a form people can easily understand—i.e., making it easier for people to structure the particular space (Raubal *et al.* 1997). It is further suggested that fewer decision points on any route and redundancy in wayfinding information are also facilitating effects.

The grouping of similar destinations into destination zones plays an important role for our case study: Gates are grouped within gate areas, for example, the gates C51, C52, C53, etc. are all part of gate area C. The agent uses such hierarchical knowledge to decide if its goal information matches any of the perceived information from signs (see section 7.4).

3.3 Computational models for wayfinding

Cognitively based computational models generally simulate a wayfinder that can solve route-planning tasks with the help of a cognitive-map-like representation (section 3.2.3). The focus of these models is to find out how spatial knowledge is stored and used, and what cognitive processes operate upon it. One can distinguish between computational process models where cognition is conceptualized as sets of rules acting on symbolic representations, and biologically inspired models that model cognition through the use of lower level, physiologically plausible mechanisms.

The TOUR model is considered the starting point for a computational theory of wayfinding (Kuipers 1978). It is a model of spatial knowledge whose spatial concepts are based mainly on observations by Lynch (1960) and Piaget and Inhelder (1967), and interviews by its designer. With the TOUR model Kuipers simulates learning and problem solving while traveling in a large-scale urban environment. His main focus of attention is the cognitive map in which knowledge is divided into five categories: (1) routes, (2) a topological street network, (3) the relative position of two places, (4) dividing boundaries, and (5) containing regions. This knowledge is represented through environmental descriptions, current positions, and inference rules that manipulate them. Routes are described as sequences of *View-Action* pairs. Because TOUR copes with incomplete spatial knowledge of the environment, it learns about it by assimilation of observations into the given structure. A subsequent application to the TOUR model utilizes an approach to robot learning based on a hierarchy of types of knowledge of the robot's senses, actions, and spatial environment (Kuipers *et al.* 1993).

Several other cognitively based computational models, such as TRAVELLER (Leiser and Zilbershatz 1989), SPAM (McDermott and Davis 1984), and ELMER (McCalla *et al.* 1982), simulate learning and problem solving in spatial networks. The program ARIADNE (Epstein 1997) learns facilitators and obstructers for pragmatic two-dimensional navigation. NAVIGATOR (Gopal *et al.* 1989; Gopal and Smith 1990) integrates concepts from both cognitive psychology and artificial intelligence. It represents basic components of human information processing, such as filtering, selecting, and forgetting. In this model, cognitive processes relating to spatial learning and using such knowledge for navigation complement two views of a suburban environment—an objective one and a subjective or cognitive one. The cognitive map is modeled through a hierarchical network consisting of nodes, links, subnodes, and sublinks. The computational process model analyzes retrieval of spatial knowledge and wayfinding by using the following measures: see if the goal is reached; amount of time taken to reach the goal; quality of pattern matching between information in memory and goal information; and errors in navigation and search strategies. O'Neill (1991) presents a model for spatial cognition and wayfinding that is built upon the biological approach. NAPS-PC (Network Activity Processing Simulator-'PC' microcomputer version) builds an artificial neural network of choice points and connecting paths from a textual list of places, preserving their topological relationships. A search for a route starts by stimulating the start and goal nodes to their maximum activity. The activity propagates from these two nodes through the network until it intersects. Nodes in-between that get activated function as subgoals during the search. The simulation models wayfinding processes of people who already have knowledge of the environment.

The focus of these computational models lies primarily in the creation and exploration of the cognitive map; they largely neglect the processes of how people immediately perceive and assign meaning to their spatial environments as they navigate through them. For example, Kuipers' TOUR model completely ignores sensory impressions. Golledge (1992) mentions the possibility of spatial knowledge not being well described by existing theories or models of learning and understanding and, therefore, calls for more research on human understanding and use of space.

3.4 Artificial intelligence and mobile robots

Artificial intelligence (AI) tries to build and understand intelligent entities (Russell and Norvig 1995). On the one hand, it strives to investigate about the human mind by creating systems that think and act like humans. On the other hand, it builds computational machines that automate intelligent behavior independent of the way humans think. AI researchers have proposed many definitions of intelligence, the one by Alan Turing based on the Turing Test being the most prominent[3] (Turing 1950).

Since the agreement to adopt John McCarthy's term *Artificial Intelligence* in 1956 different methodologies have developed within this field of research. *Symbolic AI* or *cognitivism* is built upon the assumption that intelligent behavior can be realized through the manipulation of symbols. Reasoning is therefore based on the ability to represent the world symbolically and operating rules on data structures. The General Problem Solver (GPS) by Newell and Simon was an early success of a program using the symbolic AI approach. It was designed to imitate human problem-solving, considering subgoals and possible actions (Newell and Simon 1988).

The idea of *sub-symbolic AI* or *connectionism* is that intelligence emerges from interactions of many simple processing units—i.e., neurons. This approach uses artificial neural networks that are based on the structure of nervous systems in humans and animals. It is argued that many problems, such as recognizing a face, cannot be solved by applying a sequence of rules. Sub-symbolic AI is strongly focused on learning, which is done by changing weights in a network's configuration.

Both the symbolic and sub-symbolic approach to AI have been assessed critically because they view cognition independently from the body and they lack the idea of environmental interaction. Brooks (1991a) criticized that many representations of knowledge are ungrounded and argued in favor of the world as its own best model. The paradigm behind *behavior-based* or *embodied AI* is based on situatedness and physical grounding. Intelligence is established through dynamic interactions with the world. Examples for this approach are various mobile robots such as Cog (Brooks *et al.* 1998), which have been developed at the MIT AI Lab (Figure 13). They are based on the

[3] During this test a computer is interrogated by a human via a teletype. The computer passes the test if the interrogator cannot tell if he is communicating with a computer or another human.

subsumption architecture (Brooks 1991b), where different behaviors can fire simultaneously and are ordered in a hierarchy.

The agent-based wayfinding model developed in this work is semantically grounded in people's real-world experiences: There is a direct connection between the meaning of simulated operations of the agent and actions taken by people in the real world. The agent is not physically situated in the real environment. This is not possible in our case because we also want to test buildings for ease of wayfinding before they are actually constructed. Therefore we integrate a *Sense-Plan-Act* (SPA) approach within the simulation framework.

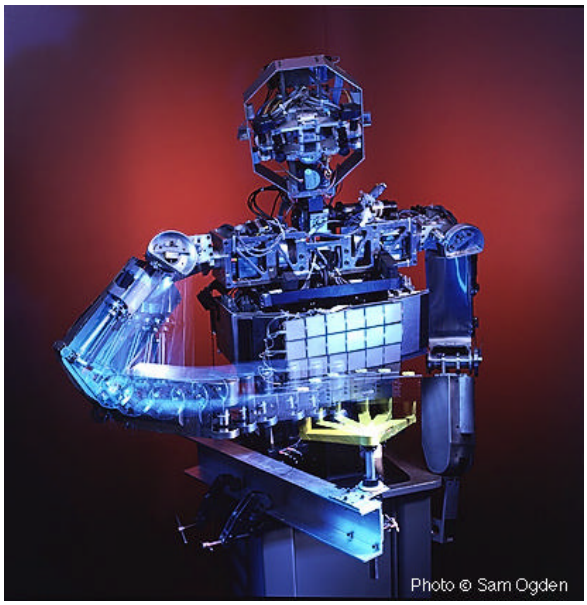

Figure 13: The humanoid robot Cog (MIT AI Lab).

This approach is based on a linear and unidirectional flow of information from sensors to world state to plan to effectors. The execution of a plan to reach a goal is analogous to executing a computer program. The three shortcomings of the SPA approach mentioned by Gat (1998) do not apply to the work done in this thesis as can be seen in Table 2.

Shortcomings of SPA approach	*Agent-based wayfinding simulation*
World may change during planning process.	Testing of proposed signage in a building is done with the assumption of a static environment.
Unexpected outcome from execution of a plan step may lead to inappropriate context for subsequent plan steps.	Simulation of the agent's behavior is guided by rules. The main reason for using the simulation is to find unexpected outcomes due to wayfinding problems.
Does not use the world as its own best model.	The building can only be simulated because it may not yet exist in reality.

Table 2: Shortcomings of the SPA approach related to the work in this thesis.

3.5 Summary

The purpose of this chapter was to describe previous work on modeling human wayfinding and connect it to the agent-based model developed in this thesis. A cognitively plausible model needs to integrate elements of human spatial cognition. Human wayfinding is a complex activity involving various cognitive and behavioral abilities. During the process of spatial reasoning people apply different decision-making methods. Two of them are integrated in our model of the cognizing agent. People cannot refer to previously acquired mental representations during wayfinding in an unfamiliar environment. The cognizing agent uses therefore only short-term representations of perceived knowledge.

In addition to empirical studies on human wayfinding performance many computational wayfinding models have been built. They are primarily concerned with the investigation of the representational aspects of knowledge. It has therefore been stated that more research on human understanding and use of space is needed. The goal of artificial intelligence is building and understanding intelligent entities. The agent-based model developed in this work is built upon the Sense-Plan-Act approach coming from this area of research.

CHAPTER 4

MODELING CONCEPTS

This chapter explains the concepts behind the model for perceptual wayfinding developed in this thesis. Our model is agent-based; therefore we introduce agent theory and describe abstract models of agents and their environments. The model also integrates elements of human perception and cognition. We explain the theory and assumptions behind Gibson's affordances and relate them to information for wayfinding. The interplay between affordances and information is at the core of the perceptual wayfinding model. At the end of this chapter Neisser's schemata are discussed.

4.1 Agents

Various definitions of what an agent is can be found in the literature. An agent can be seen as a technical concept, a metaphor, or a design model (Gilbert *et al.* 1995; Nwana and Ndumu 1996). In this thesis we use *agent* as a conceptual paradigm for the simulation of people's wayfinding behavior in unfamiliar buildings.

Not only have agents been dealt with in artificial intelligence, they have also been used to define this area of research: "AI is the study of *agents* that exist in an environment and perceive and act." (Russell and Norvig 1995, p. 1) In general, an agent can be anything that can perceive its environment through sensors and act upon that environment through effectors (Figure 14) (Russell and Norvig 1995). According to Wooldridge (1999, p. 29)

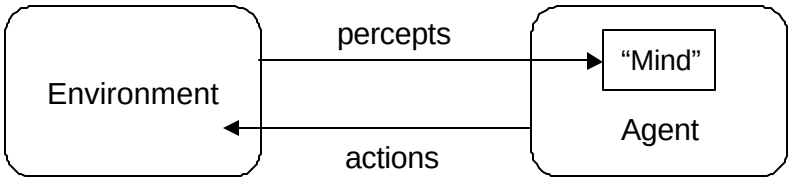

Figure 14: Agents interact with their environment—based on (Russell and Norvig 1995, p. 32).

"an *agent* is a computer system that is *situated* in some *environment*, and that is capable of *autonomous action* in this environment in order to meet its design objectives." He further states that *intelligent agents* "must operate robustly in rapidly changing, unpredictable environments, where there is the possibility that actions can fail." Therefore, intelligent agents must be capable of *flexible* autonomous actions. Flexibility means that the agent reacts to changes in the environment, exhibits goal-directed behavior, and has some sort of social ability—it can communicate with other agents in a *multi-agent system* (Weiss 1999). It is a difficult task to design agents that balance effectively between goal-directed and reactive behavior. This is important when the environment changes during the performance of a task.

There is a difference between agents and objects in the computational sense. Wooldridge (1999, p. 34) defines objects as "computational entities that *encapsulate* some state, are able to perform actions, or *methods* on this state, and communicate by message passing." Agents have a stronger notion of autonomy, because objects do not themselves have control over whether one of their methods to act is executed or not. Objects also lack flexible behavior. Agents must be distinguished from expert systems as well. Expert systems are disembodied—they do not interact directly with an environment but give advice to a third party, lack the ability to cooperate with other agents (Wooldridge 1999), and do not learn (Nwana and Ndumu 1996).

4.1.1 Abstract models of agents

In general, agents are represented as functions that map percepts to actions. A *standard agent* (Wooldridge 1999) can be abstractly seen as a function

$$action : S^* \rightarrow A \quad S = \{s_1, s_2, \ldots\},\ A = \{a_1, a_2, \ldots\}$$

that maps sequences of environment states S (S^* represents such a sequence) to actions A. Such agent decides upon an action based on its experiences, i.e., a sequence of environment states.

Wooldridge (1999) distinguishes between three different types of abstract agent models: purely reactive agents, agents with subsystems for perception and action, and agents with state.

Purely reactive agents respond directly to their environment and make decisions without reference to the past. They can be represented by the function

$action : S \rightarrow A.$

For example, a thermostat can be seen as a purely reactive agent. Its environment can be in one of two states, either too cold or temperature OK. Its actions are to turn the heater on or off.

One can separate the agent's decision function into *perception and action subsystems*. The agent's ability to perceive its environment is expressed through the function

$see : S \rightarrow P \qquad S = \{s_1, s_2, \ldots\},\ P = \{p_1, p_2, \ldots\}$

that maps environment states S to percepts P. Sequences of percepts (i.e., P^*) are then mapped to actions:

$action : P^* \rightarrow A \quad P = \{p_1, p_2, \ldots\},\ A = \{a_1, a_2, \ldots\}.$

Finally, one can model *agents with state*. For a state-based agent, perception is again represented by the function

$see : S \rightarrow P \qquad S = \{s_1, s_2, \ldots\},\ P = \{p_1, p_2, \ldots\}$

but the selection of an action is now defined by the function

$action : I \rightarrow A \quad I = \{i_1, i_2, \ldots\},\ A = \{a_1, a_2, \ldots\}$

that maps internal states of the agent to its actions. The additional function

$next : I \times P \rightarrow I$

is used to update the agent's internal state: An internal state and a percept are mapped to a new internal state. A state-based agent starts with an initial internal state, observes its environment state, and generates the `see` function. Then the agent's internal state is updated with the `next` function and an `action` selected. After performance of this action, the agent enters another such sequence.

4.1.2 Implementations of abstract agent models

The abstract agent models described in section 4.1.1 can be implemented in different ways, depending on how the decision-making of the agent is realized (Bryson 2000). Possible realizations are based on logical deduction, through a direct mapping from situation to action, they can be influenced by beliefs, desires, and intentions of the agent, and they can be implemented through various sof tware layers (Wooldridge 1999).

Russell and Norvig (1995) differentiate between four types of agent programs. *Simple reflex agents* operate based on condition-action rules. For example, in a driving agent, the

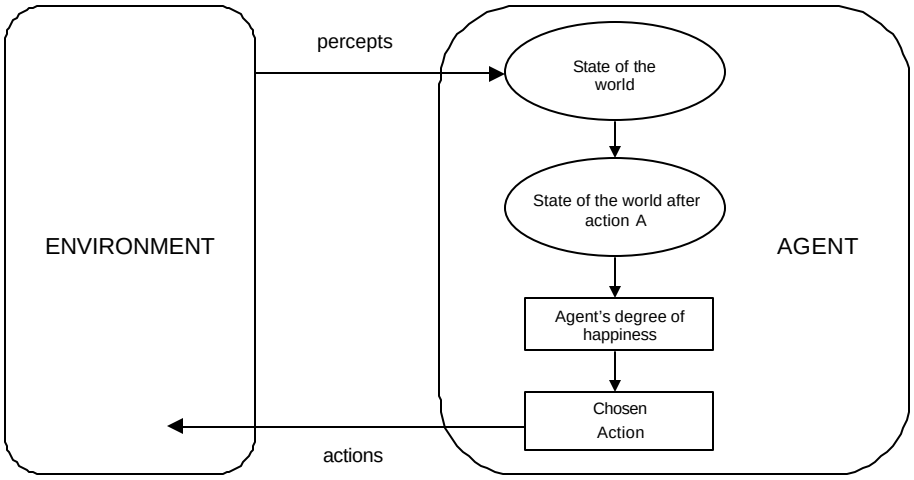

Figure 15: Structure of a utility-based agent—based on (Russell and Norvig 1995, p. 44).

condition "red traffic light" triggers the action "stop at traffic lights." These agents can be implemented very efficiently but work only for a limited range of applications.

Agents that keep track of the world use internal states to choose an action. They need information about how the world evolves through percepts and also knowledge about how its actions affect the world.

Goal-based agents need current state descriptions and goal information describing which situations are desirable. The agent combines these with information about results of possible actions and then chooses an action to achieve its goal. Goal-based agents are more flexible with respect to achieving different goals: If a new goal is specified then the agent comes up with new behavior.

In many situations there are different ways to reach a goal. For example, a driving agent can use different roads to go from A to B, but some are more reliable than others. *Utility-based agents* (Figure 15) have valuation functions, which allow them to compare between different action sequences to achieve a goal. Such functions map world states or sequences of world states to real numbers, which describe the associated degrees of happiness.

Before designing an agent program, one needs to know both about the possible percepts, actions, and performance measures of the agent, and about the kind of environment in which it will act. The PAGE (Percepts, Actions, Goals, Environment)

Agent type	*Percepts*	*Actions*	*Goals*	*Environment*
Satellite image analysis system	Pixels of varying intensity and color	Print categorization of scene	Correct categorization	Images from orbiting satellite
Refinery controller	Temperature and pressure readings	Open and close valves, adjust temperature	Maximize purity, yield, and safety	Refinery

Table 3: Two PAGE descriptions from (Russell and Norvig 1995).

description is useful in this respect, illustrating possible percepts, actions, and goals of an agent in a specific domain. Table 3 shows two examples for such a description (Russell and Norvig 1995, p. 37).

4.1.3 Environment of an agent

Agents have to be coupled with an environment in which they perceive and act. The nature of this connection is the following: The environment provides percepts to the agent, the agent decides upon and performs actions in—and therefore on—the environment, which in turn provides new percepts, etc. The complexity of this process is influenced by the properties of the environment. Russell and Norvig (1995) distinguish between the following different properties of environments:

- *Accessible* versus *inaccessible*
  The environment is accessible to the agent if the agent's sensory apparatus gives it access to the complete, accurate, and up-to-date state of the environment. Most environments people deal with on a day-by-day basis such as the everyday physical world are inaccessible.
- *Deterministic* versus *nondeterministic*
  The environment is deterministic if its next state is completely determined by the current state and the actions selected by the agent. The everyday physical world is therefore nondeterministic.
- *Episodic* versus *nonepisodic*
  In an episodic environment the agent's experience is divided into episodes. Within each episode the agent perceives and acts. Subsequent episodes do not depend on

performed actions in previous episodes. Car driving is nonepisodic, whereas an image-analysis system works in episodes.

- *Static* versus *dynamic*
  If the environment can change while the agent is deciding on an action, such as in the everyday physical world, then it is dynamic.
- *Discrete* versus *continuous*
  The environment is discrete if there are a limited number of distinct percepts and actions. For example, chess is discrete, whereas car driving as most of the everyday world is continuous.

The environment can be modeled through a state—an initial state at the beginning—and an update function. It is updated based on the agents' actions and other processes in the environment. If the agent model is based on states, then these states must be seen apart from the states of the environment because the states of an individual agent are constructed from its percepts alone. The agent does not have access to the complete state information of the environment.

4.2 Affordances

Affordances originate from the ecological approach to science. The concept was originally developed by Gibson and gave rise to later criticism mainly because it is based only on perception and ignores processes of cognition.

4.2.1 The ecological viewpoint

The ecological approach to psychology was developed to solve the major problem of cognitive psychology, i.e., the problem of knowledge. It is based on ecological science, a multidisciplinary advance to the study of living systems, their environments, and the reciprocity between the two. Ecological psychology proposes to study the information transactions between living systems and their environments, especially with regard to the perceived significance of environmental situations for the planning and execution of purposeful behaviors. The world is seen as the information source for perception and action.

The ecological approach is strongly opposed to the information-processing framework that is founded on a human-machine analogy. Shaw and Bransford (1977) argued that humans and machines are different in many aspects such as emotion, personality, social factors, and culture. Humans are also active and intent-driven; therefore one cannot compare such active, knowledge-seeking beings with unconscious, static machines that lack any trace of natural motivation. Ecological psychology denies that nature communicates to us in the form of data inputs that must be translated by a phalanx of cognitive homunculi into a more readable form: We extract meaning directly through our perceptual systems, therefore knowing is a direct process. Ecological theory suggests that the perceptual system extracts invariants embodying the ecologically significant properties of the perceiver's world. If the senses function reliably, then perceptual information specifies true propositions about the world.

4.2.2 Gibson's affordances

James J. Gibson investigated how people visually perceive their environment and thereby coined the term *affordance* (Gibson 1977; 1979). According to Gibson the environment consists of a medium, substances, and surfaces. We move in a *medium*—of light, sound, odor, etc.—in which there are points of observation and lines of locomotion. The *substances* differ in chemical and physical composition, and are structured in a hierarchy of nested units. The medium is separated from the substances of the environment by *surfaces*. An important point in Gibson's theory is that animal and environment are an inseparable pair. Species of animals occupy *ecological niches*—settings of environmental features that are suitable for an animal. An ecological niche implies a certain kind of animal and an animal implies a certain kind of ecological niche. This complementarity is not implied by classical physics, therefore Gibson uses the principles of *ecological physics* to describe the properties of substances and surfaces. Such physics considers functions of the environment at an ecological size level in contrast to a description in terms of space, time, matter, etc., within classical physics. This approach seems to be superior for the study of people's everyday perception and behavior. Gibson argues that "the fundamental ways in which surfaces are laid out have an intrinsic meaning for behavior unlike the abstract, formal, intellectual concepts of mathematical space." (Gibson 1979, p. 44)

Gibson describes the process of perception as the extraction of invariants from the stimulus flux. Surfaces absorb or reflect light and Gibson's radical hypothesis is that the composition and layout of surfaces constitute what they afford. Affordances are therefore specific combinations of the properties of substances and surfaces taken with reference to an observer[1]. These invariant compounds are specified in ambient light—which is the result of illumination—and detected as units. Ambient light has structure and therefore information. This is at the heart of ecological optics, which is concerned with available information for perception. Gibson further argues that it is easier to perceive invariant units than to perceive all the variables separately.

The theory of affordances is influenced by Koffka's (1935) work on Gestalt psychology, where he states that "each thing says what it is." Gibson argues that by looking at objects, people perceive their affordances and not their physical qualities, such as size or color, as proposed by orthodox psychologists. In addition, the whole realm of social significance for human beings can be described by what other persons afford.

As already mentioned, affordances have to be described relative to the person. For example, the affordance "to sit" is relative to the size of an individual. Much later work with affordances builds on this fundamental tenet of ecological psychology, called *agent-environment mutuality* (Gibson 1979; Zaff 1995). This suggests that at a fundamental level various aspects of agents and their environment need to be understood in terms of the relationships between them. Neither can be modeled without reference to the other. According to Zaff (1995) affordances are measurable aspects of the environment, but only to be measured in relation to the individual. Particularly, it is important to understand the *action relevant* properties of the environment in terms of values intrinsic to the agent. For example, Warren (1995) shows that the "climbability" affordance of stairs is more effectively specified as a ratio of riser height to leg length (R/L) (Figure 16). Experimentally, subjects of different heights perceived stairs as climbable depending on their own leg length, as opposed to some extrinsically quantified value. A ratio of 0.88 (R/L) was found to be the critical point where subjects, regardless of height, shifted their estimate from climbable to not climbable. Other low-level affordances for objects, including object height, "sittability", and "graspability" have been studied to determine similar body-scaled ratios (Mark 1987; Bingham and Muchisky 1995; Warren 1995).

[1] Gibson therefore suggests that an ecological niche is a set of affordances.

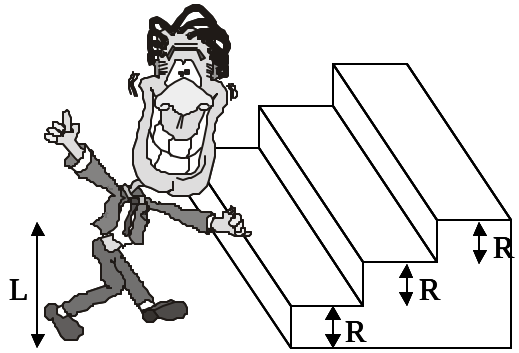

Figure 16: The "climbability" affordance of stairs specified as ratio of riser height to leg length (R/L).

Additionally, dynamic or task specific conditions must be considered. In his discussions of walking through apertures Warren (1995) points out the necessity of such considerations. Anatomical measurements of individuals cannot simply be matched with door dimensions to determine aperture "passability." The act of walking produces movement that impacts one's ability to pass through a door, and accordingly to perceive this affordance. It's likely that other dynamic factors such as walking speed would also impact the perception of "passability."

4.2.3 Deficiencies and further work on affordances

Many researchers believe that Gibson's theory is insufficient to explain higher-order processes such as wayfinding (Montello 2000) because it is grounded only on perception and neglects processes of cognition. Neisser (1976) argues that Gibson's theory does not say what kinds of cognitive structures perception requires. It does not explain what happens when we choose what to see and how errors are possible. Furthermore, it seems unclear how people perceive things without a coupling to the environment—e.g., processes concerning memory and thought. Eco proposes that Gibson's "fundamentally realistic and nonconstructivist" (Eco 1999, p. 203) ecological theory of perception needs to be supplemented by the notion of *perceptual judgments*, that is, by applying a cognitive type—something that permits recognition—and integrating the stimuli with knowledge from previous experiences. Lakoff (1987, p. 216) states that "the Gibsonian environment is not the kind of world-as-experienced that is needed in order to account for the facts of

categorization … his account only deals with *individual* phenomena, not *categories* of phenomena."

Norman (1988) investigated affordances of everyday things, such as doors, telephones, and radios, and argued that they provide strong clues to the operation of such things. He adapted Lakoff's view and recast affordances as the results from the mental interpretation of things, based on people's past knowledge and experiences, which are applied to the perception of these things. Furthermore, Gaver (1991) stated that a person's culture, social setting, experience, and intentions also determine her perception of affordances. Affordances, therefore, play a key role in an *experiential* view of space (Lakoff 1988; Kuhn 1996a), because they offer a user-centered perspective. Similarly, Rasmussen and Pejtersen (1995) pointed out that modeling the physical aspects of the environment provides only a part of the picture. "The framework must serve to represent both the physical work environment and the 'situational' interpretation of this environment by the actors involved, depending on their skills and values." (Rasmussen and Pejtersen 1995, p. 122) This can be broken into three relevant parts, the mental strategies and capabilities of the agents, the tasks involved, and the material properties of the environment.

4.3 Affordances and information for wayfinding

Both affordances and information are essential for people finding their ways in an unfamiliar environment. Affordances suggest possibilities for behavior and information helps them to choose between alternatives. In this sense we propose that decision-making during wayfinding in an unfamiliar building is based on the interaction of affordances and information.

4.3.1 Affordances for wayfinding

Environments offer vast quantities of affordances to their users. According to Kuhn (1996a) spatial affordances can be grouped into four categories reflecting different task situations. These categories are shown in Table 4, in addition with generic examples and specific examples from our case study.

Affordances for	*Example*	*Example from case study*
individual user	move	move from check-in counter to gate
user and individual entity	objectify	perceive and interpret sign
user and multiple entities	differentiate	differentiate gates
groups of users	communicate	communicate with other people in the airport

Table 4: Categories of affordances according to Kuhn (1996a).

When performing a wayfinding task in a spatial environment people utilize a set of affordances. Some of them are involved in the control of locomotion such as moving along a hallway, others are a prerequisite for information acquisition such as reading and interpreting different gate signs, etc. The most prominent affordances during wayfinding are the "move" or "go-to" affordances offered by paths. The articulation of paths is a fundamental aspect of wayfinding communication because they indicate directions of movement (Arthur and Passini 1992). In general, a path affords locomotion from a start point to an end point (Figure 17), between features that prevent locomotion—for example, obstacles such as columns or a wall (Figure 18). Paths, which are clearly discernible—through markings on the ground or guiding structures on the side or above—facilitate visually controlled locomotion, which is directed by visual perception and depends on sequential optical information (Heft 1996).

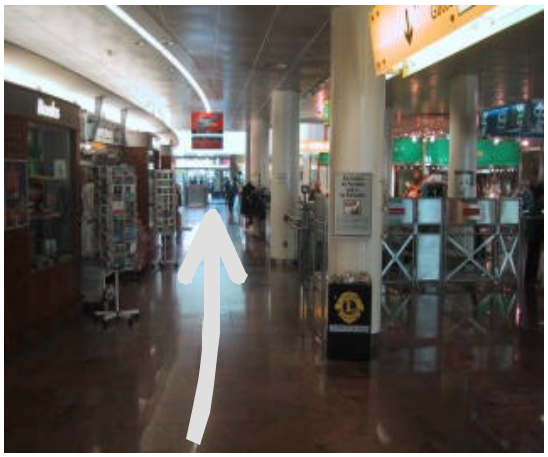

Figure 17: "Go-to" affordance of a path.

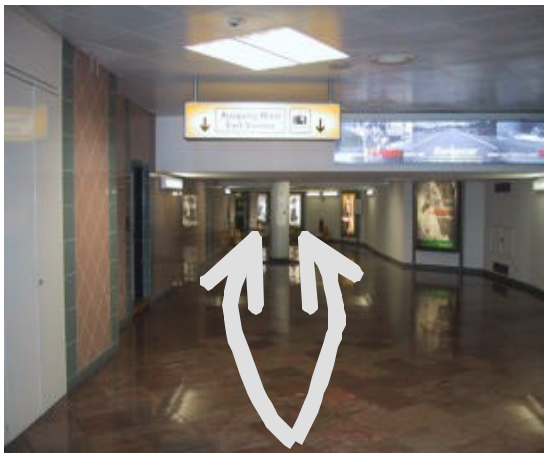

Figure 18: Paths afford moving between obstacles.

4.3.2 Information for wayfinding

Wayfinding requires more than the ability to avoid collision with other objects; in unfamiliar environments it requires guidance information for the wayfinder (see section 3.2). Such information can be communicated through different means, such as architectural features—e.g., columns forming the boundary of a pathway—, lighting—e.g., as directional guidance—, and signs, which are the focus of our case study[2].

In general, *information* is what is different from random noise (Shannon and Weaver 1949). It can be discovered where causes leave effects (Pinker 1997). People use information to make decisions, information is therefore answering people's questions. In this sense, the meaning of information is defined through the decisions and actions for which it is used. In particular, *geographical information* is any information about a spatial situation (Frank *et al.* 2000). It is needed for two types of decisions: location and allocation of resources, and wayfinding (Frank forthcoming-a). Wayfinding signs offer geographical information to wayfinders because they indicate properties, which are found at a given location. They are used to convey general information about a setting, communicate directions to destinations, and identify destinations (Arthur and Passini 1990). Thereby, they answer questions such as "where is a certain object?" (e.g., gate C 54) and "where are all objects with certain properties?" (e.g., all gates belonging to gate area C).

4.3.3 Wayfinding as an interplay of affordances and information

We propose that decision-making during the performance of a wayfinding task is based on the interplay of affordances and information. Figure 19 shows a scenario where a wayfinder perceives two "go-to" affordances, i.e., the affordance of following the hallway to the left and the affordance of following the hallway to the right. In order to choose the correct continuation according to a given task—which will result in locomotive behavior—the wayfinder needs additional information on where these paths lead. In this case, one can see from the signs that the path to the left leads to "Terminal 2" whereas the path to the right leads to "Terminal 1."

---

[2] Handicapped people might depend on different modes of communication. For example, auditory sources are an important aspect for communicating wayfinding information to blind people (Raubal and Frank 2000).

Figure 19: “Go-to” affordances and the related sign information.

The general framework is the following: If the wayfinder first perceives information at a decision point, then she acquires a particular piece of information. This piece of information can only be used for deciding upon an action if the wayfinder also perceives the related affordance. After perceiving a particular affordance, the wayfinder decides if she wants to utilize it by taking into account this piece of information (Figure 20a). In the other case, the wayfinder first perceives an affordance at a decision point and then has to

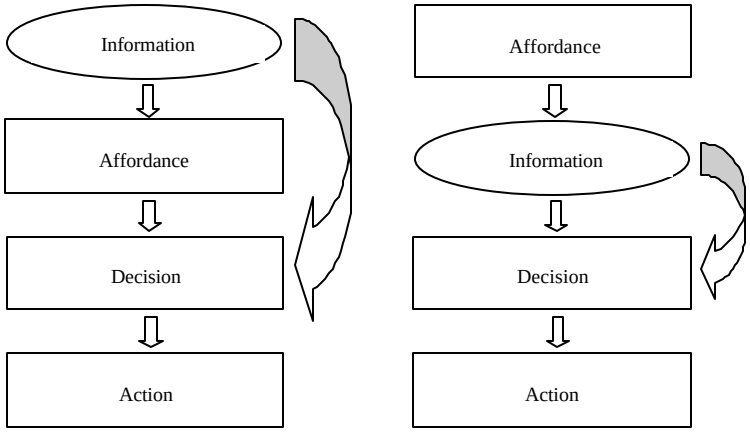

Figure 20 a, b: Perceiving information or affordances first.

pick up the related information. The resulting piece of information is again used for deciding upon an action (Figure 20b).

We account for the necessity of relating "go-to" affordances with information by modeling *spatial situations* as *affordance-information pairs*. When perceiving a "go-to" affordance, the agent is therefore also provided with the related information, such as "utilizing affordance x brings me closer to location y."

In this thesis we do not investigate the process of perception itself but use the concept of affordance to model wayfinding in the sense that affordances are what objects or things offer people to do with them. Therefore, they create potential activities for the agent. We supplement Gibson's theory with elements of cognition, situational aspects, and social constraints to compensate for the deficiencies described in section 4.2.3.

4.4 Schemata

Ulric Neisser's (1976) perceptual theory is an ecologically oriented approach that combines direct perception and schema theory, therefore integrating perception and cognition. It is based on the perceptual cycle (Figure 21), which tries to integrate the classical theories of information-processing, information pick-up, and hypothesis testing. Neisser's main argument is that we need cognitive structures to pick up information from the environment—the perception of meaning depends on schematic control of information pickup. These structures are anticipatory schemata that prepare the perceiver to accept

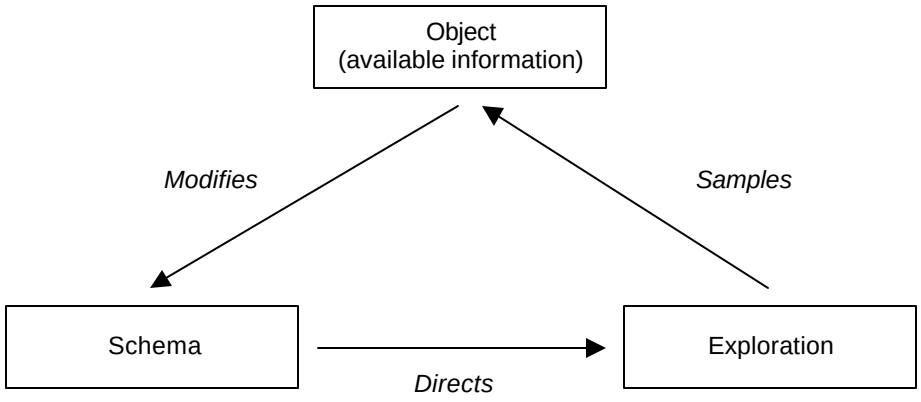

Figure 21: Neisser's perceptual cycle—based on Neisser (1976, p. 21).

certain kinds of information rather than others.

> *A schema is that portion of the entire perceptual cycle which is internal to the perceiver, modifiable by experience, and somehow specific to what is being perceived. The schema accepts information as it becomes available at sensory surfaces and is changed by that information; it directs movements and exploratory activities that make more information available, by which it is further modified (Neisser 1976, p. 54).*

Such schemata are momentary states of the perceiver's nervous system. According to Neisser's cyclic model of perception, we only pick up information for which we have a schema. This explains why we focus on some events—the events involved in the cycle of task-relevant expectations, explorations, etc.—and not others. Neisser further argues that there are also schemata that always operate—outside of our attention. These *preattentive processes* are necessary to survive.

The cyclic model of perception serves as another foundation for our proposed agent-based model for wayfinding. The agent has an internal observation schema, which includes the wayfinding goal. During the wayfinding process this schema directs the agent's processes of exploration and information pickup—the pickup of affordances and related information from the environment.

4.5 Summary

Agents come from the discipline of artificial intelligence. They can interact with their environment and are generally represented as functions, which map percepts to actions. Different abstract agent models exist. We use agent as a conceptual paradigm for the perceptual wayfinding model.

Gibson's idea of affordances is a result of taking an ecological scientific viewpoint. Affordances are what things in the environment offer us to do. Therefore they have to be taken with reference to a specific observer. Affordances are solely based on perception and their theory has been criticized for neglecting elements of cognition. In this thesis we use affordances to model the agent's epistemology—what the agent can know about its environment. We thereby add elements of cognition, situational aspects, and social constraints to Gibson's theory of perception.

The “go-to” affordance is the most important affordance during wayfinding. Its utilization leads the wayfinder along a path. In order to choose the correct path among different alternatives, these “go-to” affordances have to be connected to wayfinding information such as from signs. We therefore propose that the process of wayfinding works based on the interplay between various affordances and information.

Neisser’s schema theory can be seen as one answer to the deficiencies of Gibson’s theory of perception. It integrates elements of perception and cognition. We use it as a conceptual basis for the agent’s observation schema.

CHAPTER 5

FORMAL METHODS

In this thesis we develop a formal agent-based wayfinding model. This chapter explains the formal methods used to construct it. The parts of graph theory needed to represent the agent's environment for the simulation are introduced in the first section. For formalization we employ algebraic specifications written in a functional programming language. The second section explains the components of an algebraic system, gives an example for algebraic specifications, and demonstrates their usefulness. The final section is dedicated to functional programming languages in general and Haskell, the language used for this work, in particular. We describe the main concepts behind it and give examples of its syntax to an extent necessary to understand the specifications provided in the thesis.

5.1 Graph theory

Graphs are an important mathematical tool to model physical and virtual networks, such as road networks, utility networks, and air traffic networks. They describe relations between points of a system and offer the possibility to assign numerical parameters to the connections between the points. For example, in a road network points might represent different cities and the connections might represent distances between these cities. In this work we use graphs to represent the agent's wayfinding environment. This section gives mathematical definitions for different elements of graph theory as used in the thesis. It is based on work done by Sowa (1999), Kirschenhofer (1995), Laurini and Thompson (1992), and Piff (1991).

Formally, a *graph* G consists of a set N of *nodes* and a set E of *edges*[1]. Every edge in E is a connection between two nodes of N and represented as a pair of these nodes. It depends on the practical situation whether the order of the two nodes is important or not. If the order is irrelevant, then the graph is called *undirected* and consists of a set of unordered

[1] Different authors use different terms for nodes—points, vertices, etc.—and edges—lines, arcs, etc.

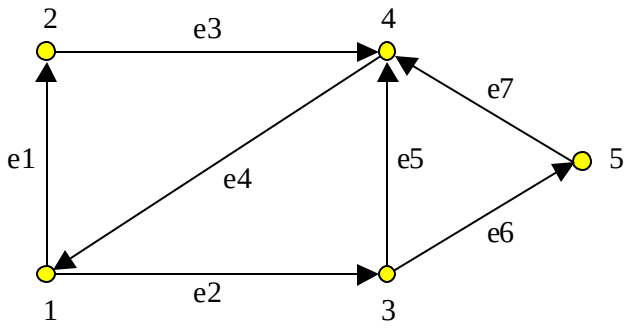

Figure 22: Directed graph.

pairs. If the order is relevant, then <*1,2*> and <*2,1*> represent distinct edges, and the graph is called *directed*. Directed edges are marked through arrowheads pointing to the second node of the ordered pair (Figure 22).

Theoretically, graphs are defined in an abstract way, but visualizing them as diagrams facilitates their readability. Many descriptive terms reflect this idea. Let *e* be the edge <*1,2*>. Then the nodes *1* and *2* are called *endpoints* of *e* and *e* is said to *connect 1* and *2*. If *e* is an edge of a directed graph, then *1* is called the *source* of *e* and *2* is called the *target* of *e*. An edge that connects a node with itself is called a loop, e.g., *e* = <*1,1*>.

There are different ways of traversing a graph, depending on the restrictions imposed on the combination of edges. A *walk* through a graph is a sequence of succeeding nodes for which any two adjacent nodes are the endpoints of some edge. If a walk contains *n*+1

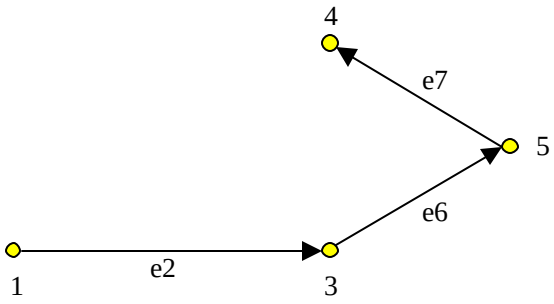

Figure 23: Direct ed path.

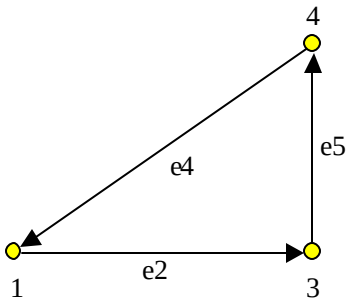

Figure 24: Directed cycle.

nodes, then it must traverse *n* edges and is said to be of *length n*. A *path* is a walk where all nodes are distinct. A walk in which the first and last node are the same, but all other nodes are distinct, is called a *cycle.*

A walk, path, or cycle through a directed graph *G* may or may not follow the direction of the arrows. We call a walk, path, or cycle *directed* if adjacent nodes occur in the same order as in some edge of *G*. For example, if *1* and *2* are adjacent nodes on a path, then the ordered pair *<1,2>* must be an edge of *G*. Edges in a directed graph are like one-way streets and a directed path follows these one-way streets. Figure 23 shows a directed path—the sequence of edges *<1,3>, <3,5>, <5,4>*—and Figure 24 a directed cycle—the sequence of edges *<1,3>, <3,4>, <4,1>*—for the directed graph given in Figure 22.

A graph *G* is called *connected* if there exists a path between any two nodes in *G*. If it is not connected then it breaks down into disjoint *components*. Each of the components is connected, but none of them is linked to any of the others through a path. A directed graph *G* is called *strongly connected* if for any nodes *a* and *b* there exists a path from *a* to *b* in *G* and also a path from *b* to *a* in *G*.

5.2 Algebraic specifications

The purpose of specifications is the mathematical description of concepts. Specifications can be informal—e.g., expressed in a natural language—or formal. Formal specifications are the link between a conceptual model and its implementation. They are used to formally prove the correctness of the latter (Liskov and Zilles 1979; Liskov and Guttag 1986). Formal specifications can be processed by a computer and are therefore more reliable than when compared with the intuitive understanding of a reader. This increases the likelihood that a program implemented based on its specifications will perform the intended functions.

Among the various existing specification methods—such as state machine models and axiomatic descriptions—algebraic definitions have proven to be good candidates for specifying data abstractions for spatial and temporal domains (Car and Frank 1995; Kuhn 1996b; Frank and Kuhn 1999; Medak 1999; Bittner 2001; Winter and Nittel forthcoming). Data abstractions are based on *abstract data types*, which are representation-independent formal definitions of all operations of a data type (Guttag *et al.* 1978). Algebraic specifications describe objects in terms of their operations. They do not express what the

objects are, but how they behave—also called the *implicit definition approach*. Algebraic specifications are based on a solid mathematical foundation, the theory of algebras, and therefore mathematical methods can be applied to them.

5.2.1 Algebra and definitions

In general, the notion of *algebra* is based on the idea that mathematical structures can be described in terms of the operations applicable to them. An algebraic system is therefore "a set of elements of any sort on which functions such as addition and multiplication operate, provided only that these operations satisfy certain basic rules." (Mac Lane and Birkhoff 1999, p. 1) From a software engineering perspective, an algebraic specification consists of three parts:

1. a set of *sorts* designating the objects involved;
2. a set of *operations* applicable to these objects; and
3. a set of *axioms* defining the behavior of these operations;

A *sort* is the name for an element or object of a particular type. Sorts are therefore used to abstract from individual values to sets of values. If the set contains sorts of only one type, then the algebra is called *single-sorted*. A *multi-sorted* algebra contains sorts of different types.

Operations are applied only to the defined sorts. One can distinguish between two kinds of operations: constructors and observers (Liskov and Guttag 1986). *Constructors* are the operations used to construct all the values of a sort. Their result is always an object of the defined sort. *Observers* are the operations used to observe properties of a sort. Their result is an object of another sort.

Axioms describe the properties of the operations by defining their behavior. A set of axioms can be seen as a set of rules that describes the effects of an operation in terms of other operations on the same sorts.

5.2.2 An example for algebraic specifications

The following example based on (Liskov and Zilles 1979) and (Frank 1999) is instructive with regard to understanding the idea behind algebraic specifications. A multi-sorted algebra is used to define the behavior of a stack.

```
Algebra Stack (stack of a, a)
Operations:  create :: stack of a                        -- constructor
             push :: stack of a -> a -> stack of a       -- constructor
             top :: stack of a -> a                      -- observer
             pop :: stack of a -> stack of a             -- observer
Axioms:      top (push s a) = a                          -- axiom 1
             pop (push s a) = s                          -- axiom 2
             top (create) = error                        -- axiom 3
             pop (create) = error                        -- axiom 4
```

This algebra consists of two different sorts: the stack as a whole (of type *stack of a*) and its elements (of type *a*). The constructor operations create an empty stack (*create*) and put an element onto a stack (*push*). The observer operations can be either used to return the top element of the stack (*top*) or to return the stack with the top element removed (*pop*). The behavior of these operations is defined by the axioms. The top element after pushing an element onto the stack is the element pushed on (*axiom 1*). The returned stack with the top element removed after having pushed an element onto the stack is the same stack as before the operation *push* (*axiom 2*). Finally, the operations *top* and *pop* on a created and therefore empty stack are not defined and result in an error (*axiom 3* and *axiom 4*). To make *top* and *pop* total functions, the set of sorts needs to be extended by adding an element *error* representing an undefined value.

5.2.3 Usefulness of algebraic specifications

Algebraic specifications satisfy the criteria of formality, constructibility, comprehensibility, minimality, wide range of applicability, and extensibility to a large degree (Liskov and Zilles 1979). They are a useful aid during the design process of reliable software (Guttag *et al.* 1978). Starting from a conceptual model one specifies abstract data types through their operations and axioms. It is possible to create complex types by using specifications of simpler types. Abstract data type specifications are representation-

independent and can therefore be used for different implementations. Each eventual implementation is an instance of the specification.

Formal specifications are also used to prove the correctness of an implementation. This is done by showing that the implementation satisfies the original axioms. For algebraic specifications in particular, this means that the implementation defines an isomorphic image of the algebra (Liskov and Zilles 1979). An isomorphism is a one-to-one or abstractly identical mapping from one algebra to another preserving the operations.

Finally, algebraic specifications can be used for early testing—testing at design time. In this sense, specifications for a software system can be tested before committing to build the system. Algebraic specifications written in an executable programming language can be tested as a prototype (Frank and Kuhn 1995; Bittner 2001; Winter and Nittel forthcoming).

We use algebraic specifications in this thesis to formalize the agent-based process model for perceptual wayfinding. This allows for checking the consistency of the conceptual model and also to generate test cases from our case study. The proposed specifications serve as guidelines for future implementation.

5.3 Functional programming and Haskell

Functional programming languages are convenient tools to express algebraic specifications because both of them use a similar syntax and have similar mathematical foundations. Haskell is used for formalization in this thesis.

5.3.1 Functional programming languages

Functional programming languages can express semantics, are easy to read and write, and permit rapid prototyping through executable specifications (Frank and Kuhn 1995). This allows showing deviations from the intended program behavior immediately and therefore to see if the program corresponds to the designer and user requirements. Functional programming languages are also extendible—functions can be combined to form algebras and one can combine different algebras (Frank 1999).

In functional programming languages, functions are the central model components and can be used as data. Programming consists of building definitions in the form of functions,

which are evaluated by a computer. Evaluation of expressions is done by *substitution* and *simplification*. The main function is defined through subsidiary functions, which are again defined through other subsidiary functions, and so on, until at the bottom level the functions are language primitives. An expression is *canonical* if it cannot be further simplified (Bird and Wadler 1988). Contrary to *structured* programming languages such as PASCAL functional programming languages do not have an explicit flow of control because they are not executed line by line.

Functions in *pure functional languages* produce only one result value and do not have *side effects*—the input parameters are not changed and only the function's arguments are being operated on at the same time. This results in *referential transparency*, i.e., an expression always produces the same result because values can only be assigned once to a parameter. In contrast, *structured* programming languages allow *destructive assignments* such as *a := a + 1*, which results in changes of the value of *a* although the parameters do not change. The previous assignment of *a* gets "destroyed" and replaced by a new one.

One important concept used to define functions in functional programming languages is *recursion*, whereby the definition of the function refers to the function itself. Such a mechanism is necessary because loop expressions as used in structured programming languages are not allowed. The principle of recursion is demonstrated by the following definition of the factorial function over natural numbers (Thompson 1999).

```
factorial (n) = if (n==0) then 1 else (n * factorial (n-1))
```

5.3.2 The functional programming language Haskell

The functional programming language used to write specifications in this thesis is called Haskell, named after Haskell B. Curry who was one of the pioneers of the λ calculus—a mathematical theory of functions (Michaelson 1989). Haskell is *purely functional*, *strongly typed*, and uses *lazy evaluation*. A variety of Haskell implementations is available; here we use the Hugs 1998 system (Thompson 1999). This section gives a short introduction to the syntax and functionality of Haskell. A detailed tutorial can be found in (Hudak *et al.* 2000).

5.3.2.1 Strong typing

A major strength of functional programming languages such as Haskell[2] is that they are *strongly typed*. This means every object has a particular type and the compiler checks that operations can only be applied to certain types. A type system prevents the occurrence of execution errors during a program's runtime. *Type inference mechanisms* allow the logical deduction of types even when little or no type information is given explicitly. For example, given two constants *x* and *y* of type *Integer*, the type of $z = x + y$ is automatically inferred as *Integer*.

Haskell uses several *predefined types* such as integers (`Int`), floating point numbers (`Float`), characters (`Char`), strings (`String`), tuples (`(a,b)`), lists (`[a]`), and Boolean values (`Bool`). The "`::`" can be read as "has type."

```
3 :: Int
3.5 :: Float
'm' :: Char
"martin" :: String
(3,'m') :: (Int,Char)
[1,2,3] :: [Int]
True, False :: Bool
```

User-defined data types are introduced with the keyword `data` and defined by the constructors of the type. In the following example, the data type `Point` is defined by applying the constructor function `Point` to two floating-point numbers. Note that the data type name and the name of the constructor function can be the same.

```
data Point = Point Float Float
```

Enumerated types consist of a finite number of values—nullary data constructors—separated by a "`|`". The data type for the four seasons serves as an example.

```
data Season = Spring | Summer | Fall | Winter
```

Haskell also provides the possibility to define *type synonyms*. These are names for commonly used types and created with a type declaration. For example, the type `Position` behaves as the predefined type `Int`.

```
type Position = Int
```

[2] LISP, for example, performs type checking during program execution and is therefore *untyped*.

Functions in Haskell are defined as a series of *declarations*. Usually, a *type signature* declaration is followed by one or more *equations*. The function `add` defines the addition of two integers.

```
add :: Int -> Int -> Int
add a b = a + b
```

The type signature defines all input parameters and the output parameter for a function. In this case, the function `add` maps two input values of type `Int` to an output value of type `Int`.

Haskell incorporates *polymorphic types*—types that are universally quantified in some way over all types, also called *parametric polymorphism*. This allows for defining functions applicable to various types. The following function `length`, used to count the number of elements in a list, demonstrates this. We can apply it to a list of integers and to a list of characters.

```
length :: [a] -> Int
length [1,2,3,4,5] = 5
length [‘m’,’a’,’r’,’t’,’i’,’n’] = 6
```

5.3.2.2 Pattern matching

Pattern matching is a useful concept in Haskell to define functions. The left-hand sides of the equation contain patterns, which are matched against actual parameters during the application of the function. The process of pattern matching is sequential. If the match of an equation succeeds, the right-hand side gets evaluated and returned as the result of the function. If the match fails, the next equation is tried, and so on. If all equations fail, the result is an error. As an example for pattern matching we use the function `length` again.

```
length :: [a] -> Int
length [] = 0
length (x:xs) = 1 + length xs
```

When applying this function, the patterns `[]` and `(x:xs)` are matched against actual parameters, whereby `[]` matches only the empty list and `(x:xs)` matches any list with at least one argument—`x` being the first argument and `xs` the rest of the list. In general, patterns can be literal values, variables, wildcards, tuples, and constructors (Thompson 1999).

5.3.2.3 Classes and instances

A typical feature of Haskell is another type of polymorphism, called *ad hoc polymorphism* or *overloading*. Overloaded functions can be used for a variety of types—with different definitions being used for different types. Overloading therefore allows for the reuse of existing function names. In Haskell, *classes* are a mechanism for assigning types to overloaded functions. A class is a collection of types over which a function is defined. For example, the equality class `Eq` contains a set of types over which the equality operator (`==`) is defined.

```
class Eq a where
   (==) :: a -> a -> Bool
```

One then needs to define the members of the class—i.e., which types are instances of the class—and the actual behavior of the equality operator on each of these types. Built-in types of `Eq` include `Int`, `Float`, `Bool`, and `Char`. Two other possibilities are instance declarations covering two- and three-dimensional points. Pattern matching is used for the definitions of equality.

```
instance Eq Point where
   (==) (Point2D x1 y1) (Point2D x2 y2) = (x1==x2) && (y1==y2)
   (==) (Point3D x1 y1 z1) (Point3D x2 y2 z2)
             = (x1==x2) && (y1==y2) && (z1==z2)
```

5.3.2.4 Lazy evaluation

Haskell uses a *lazy (non-strict) evaluation* strategy, which means that an argument to a function gets evaluated only if the argument's value is needed to compute the overall result. If the argument is structured, such as a tuple or a list, only those parts needed will be evaluated. It is therefore possible to use infinite data structures. These allow programming in a more abstract way as can be seen in the following example showing the use of an infinite list for iteration.

```
iterate :: (a -> a) -> a -> [a]
iterate f x = x : iterate f (f x)
```

The function `iterate` takes another function `(a -> a)` and the parameter `a` as input and produces an infinite list in the form of `[x, f x, …, f^n x, …]`. A special case is the function `powers` creating an infinite list of powers of an integer.

```
powers :: Int -> [Int]
powers n = [ n^x | x <- [0 .. ] ]
powers 2 = [1,2,4,8,16,32 ..]
```

The equation utilizes an expression called *list comprehension*, which is used to generate lists in Haskell. It can be intuitively read as "the list of all *n* to the power of *x* such that *x* is drawn from the infinite list *'0, 1, 2, etc.'*"[3]

One can extract finite portions from an infinite list by applying one of the predefined functions in Haskell such as `take`.

```
take 5 (powers 2) = [1,2,4,8,16]
```

5.3.2.5 Higher-order functions and function composition

Haskell incorporates *higher-order functions*—functions that use functions as arguments and return functions as a result. The `map` function is an instructive example in this respect. It takes a function and applies it to all elements in a list, such as incrementing the elements in a list as shown.

```
map :: (a -> b) -> [a] -> [b]
map f [] = []
map f xs = [ f x | x <- xs]
map (add 1) [3,4,5] = [4,5,6]
```

Another useful higher-order function is the `filter` function, which selects those elements in a list that satisfy a given property. For example, one can filter all even numbers from a list of integers.

```
filter :: (a -> Bool) -> [a] -> [a]
filter p xs = [ x | x <- xs, p x ]
filter isEven [1,2,3,4,5,6,7,8,9,10] = [2,4,6,8,10]
```

Function composition in Haskell is a way to improve the structure of a program and thus its readability. The top-level functions are often specified by composing a number of

[3] Haskell uses the built-in infinite lists [n ..], [n,m, ..] so that [0 ..] = [0,1,2,3,…].

functions together. Each part is designed and implemented separately—following a top-down approach. The output of one function becomes the input of another function, and so on, therefore the order plays an important role. The constraint by which functions can be composed is given by the signature of the function composition operator `(.)`.

```
(.) :: (b -> c) -> (a -> b) -> (a -> c)
(f . g) x = f (g x)
```

The following example increments the elements of a list and then filters the even numbers, the second function giving the same result but using the function composition operator.

```
filter isEven (map (add 1) [3,4,5]) = [4,6]
((filter isEven) . (map (add 1))) [3,4,5] = [4,6]
```

5.4 Summary

Graphs are well suited to model networks. We use them to represent the agent's wayfinding environment. Different concepts from graph theory needed for the simulated environment, such as *directed graph*, *path*, and *cycle*, were explained.

The formal agent-based wayfinding model developed in this thesis consists of algebraic specifications. Algebraic specifications are based on the mathematical concept of algebra and provide the link between the conceptual model and its implementation. They have been widely used in different domains to prove the correctness of an implementation and for early testing of a system.

Algebraic specifications can be expressed with functional programming languages. These are based on similar mathematical foundations and characterized by functions as the basic model components. In contrast to structured programming languages, functional languages do not have an explicit flow of control, do not produce side effects, and integrate the important concept of recursion. Haskell is the chosen functional programming language in this thesis. We described its characteristics by stressing the most important concepts necessary for understanding our formal model. These were strong typing, pattern matching, classes and instances, lazy evaluation, and higher-order functions.

CHAPTER 6

THE CONCEPTUAL MODEL FOR PERCEPTUAL WAYFINDING

In this chapter we develop the conceptual agent-based process model for simulating people's wayfinding behavior in an unfamiliar building. This model is based on the theory of *perceptual wayfinding*. We first give attention to ontological and epistemological concerns to assure that the agent-based system is firmly grounded. Such grounding is a necessary requirement to model the agent's behavior in a cognitively plausible way. Starting with our specific design considerations we then describe the conceptual model. It can be broken down into two parts: first, the agent and its environment and second, the perceptual wayfinding process represented within the *Sense-Plan-Act* framework.

6.1 Ontology and epistemology

Ontology and epistemology are basic concerns during the development of an agent-based system. By defining the ontology of a specific domain, one describes what is in this domain in a general way. More specifically, from an information systems and artificial intelligence perspective, ontologies are *content theories*, because they identify specific classes of objects and relations that exist in some domain (Chandrasekaran *et al.* 1999; Frank forthcoming-b). Paying attention to the epistemology allows the designer to focus on the agent's knowledge and beliefs. Both the ontology and epistemology are necessary foundations for the set-up and functioning of the agent-based system, especially for modeling the agent's processes of perception, cognition, and action in a plausible way. In this thesis we use the results of a prior study regarding people's wayfinding experiences in airports (Raubal 1997) to model the ontology and epistemology from the viewpoint of ecological science in general and based on the work of the ecological psychologist J. J. Gibson in particular (section 4.2).

6.1.1 Ontological concerns: the wayfinding environment

The philosophical subfield of *ontology* is defined as the science of existence. This science tries to determine "the various types and categories of objects and relations in all realms of being" (Smith 2001, p. 79). By defining the ontology for a specific domain or microworld—such as Hayes' (1985b) ontology for liquids—we describe *what is* in this domain in a general way. This results in an abstract description of the content and rules of behavior of this part of the physical world, and the linkage between the physical and conceptual world (Smyth 1992). Davis (1990) calls the ontology of a microworld its entities, its relations, and the rules that govern them.

In *Objects and Their Environments: From Aristotle to Ecological Ontology*, Smith (2001) argues that ontologists have not put much effort into describing an ontological theory of people's everyday objective environments. He therefore advocates an ontology of behavioral environments, which is made up of Aristotelian *substances* (e.g., persons, objects) and *accidents* (e.g., actions, processes), and determined by people's everyday perceptions and actions. In such environments there are substances of different sorts, such as buildings and rooms, which have stable relations with each other, and also so-called *physical-behavioral units* (Barker 1968). Physical-behavioral units are recurrent types of settings that serve as the environments for the everyday activities of persons and groups of persons—e.g., Martin's drive to work. These units are composed of people behaving in certain ways (e.g., sitting, driving) and objects (e.g., cars, streets) needed for such behavior. The idea of a physical-behavioral unit goes hand in hand with the concept of an ecological niche (section 4.2.2). As Smith (2001, p. 90) puts it: "The relation between participant and setting is to different degrees one of reciprocal co-determination."

When describing the ontology of our case study environment, we start with the assumption that the scenario of people finding their way in a building can be seen as a physical-behavioral unit. In this sense, one might compare the ontological marks of the airport microworld with the ontological marks of environments put forward by Smith (2001, p. 80) based on Aristotle's ontological marks of substances (Table 5).

Ontological marks of environments / niches / settings	*Ontological marks of the airport microworld*
Complexes of substances and accidents that require support from *participant* substances.	Wayfinders participating in various processes involving hallways, signs, etc.
Can be sustained by distinct participant substances at different times.	Wayfinders are usually distinct persons at different times.
Unity of a living thing, being neither too small nor too large.	Unity of pulsating airport microworld.
Complete determinate boundary—some objects fall clearly within it, other objects fall clearly outside it.	Airport buildings have spatial boundaries—hallways, signs, etc. fall within it; highways, gas stations, etc. fall outside it.
Actual parts which are also environmental settings.	For example, the duty-free area is a part of the physical-behavioral setting of the airport.
May be part of larger environmental setting, may be scattered through space.	For example, airport consisting of different terminals.
Takes up space, occupies physical-temporal locale, and has spatial parts.	Airport microworld is spatially extended and has divisible bulk.
Has beginning and end but is not self-identical from beginning to end.	Construction (opening) of the airport microworld, dismantling (closing) of it. Accidents have temporal parts—e.g., first wayfinding leg, second wayfinding leg.
Existence through time need not be continuous.	For example, temporary closing of airport microworld due to renovation.
There are no punctual existing environments.	Airport microworld exists over a time period.

Table 5: Comparing ontological marks of the airport microworld with ontological marks of environments / niches / settings.

With the example of our case study wayfinding in airports we demonstrate how both the ontology and epistemology (focusing on the wayfinding agent, see section 6.1.2) can be modeled based on ecological concepts. Following Gibson (1979), we subdivide the wayfinding environment into the medium, the substances, and the surfaces.

6.1.1.1 The medium

People move in a *medium*, which is for light, sound, and odor coming from different sources in the environment. In such a medium there are points of observation and lines of locomotion. During wayfinding in an airport, passengers move along such lines of locomotion and occupy different points of observation where they gather information about the environment. The absolute reference axis within the medium is defined by gravity, namely up-down.

6.1.1.2 The substances

The *substances* differ in chemical and physical composition, and are structured in a hierarchy of inter-nested units. In order to arrive at the ontology, we extracted substances (i.e., nouns) from interviews, in which people described their experiences during wayfinding in airports (Raubal *et al.* 1997). Synonyms were then merged and categories of substances formed. This method is based on *ontologies from texts* (Kuhn 2000).

Figure 25 gives the taxonomic tree of substances in an airport. It is based on *is-a* relations, which allow making transitive inferences. We can infer from "a traveler is a cognizing agent" and "a cognizing agent is a substance" that "a traveler is a substance." One has to distinguish here between *intensional definitions*—category names chosen by the constructor of the ontology—and *extensional definitions*—names used by interviewees

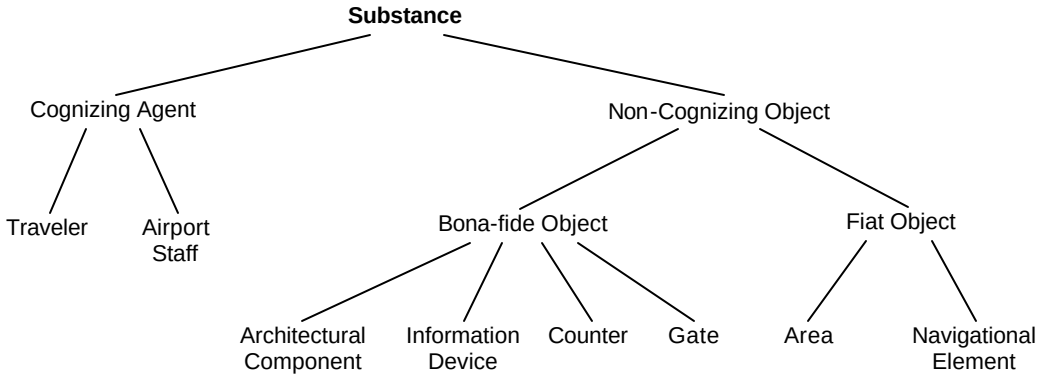

Figure 25: Intensional category definitions in a taxonomic tree of substances in an airport.

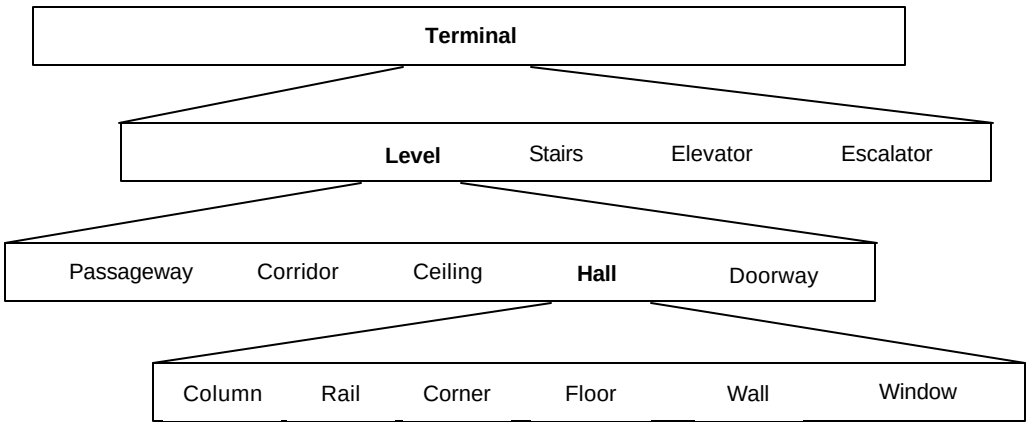

Figure 26: Partonomy of the architectural component *Terminal* in an airport.

to refer to specific instances of categories. We differentiate between two main categories of substances, which can be divided further: *cognizing agents* and *non-cognizing objects*. This is similar to Wordnet's division of the top-level category *thing* into the subcategories *living* and *nonliving* (Miller 1990). Cognizing agents in an airport can be travelers or airport staff (with instances ticketing agent, check-in agent, etc.). We further divide non-cognizing objects into *bona-fide* and *fiat objects*, depending on whether the boundaries of the objects exist independently of or are created by human cognitive acts (Smith 1995). Sub-categories of bona-fide objects in airports are *architectural component*, *information device* (e.g., sign, monitor), *counter* (e.g., check-in, passport control), and *gate*. Sub-categories of fiat objects are *area* (e.g., waiting area, gate area) and *navigational element* (e.g., path, decision point).

In addition to this taxonomy, the ontology also comprises partonomies, which are hierarchies based on *part-of* relations (Tversky 1990). Figure 26 shows examples of the partonomy for *terminal*, which is an *architectural component*. All the elements of this partonomy have physical boundaries and serve as receptacles for people's projections of fiat boundaries.

6.1.1.3 The surfaces

The medium is separated from the substances of the environment by *surfaces*. Gibson argues that the layout of surfaces has an intrinsic meaning for behavior. This will be dealt with in the next section.

6.1.2 Epistemological concerns: the agent

The epistemological question of what the wayfinding agent can know about the environment and how it can accumulate such knowledge is modeled through affordances (section 4.2). According to Gibson, the composition and layout of surfaces constitute what they afford. Affordances are specific combinations of the properties of substances and surfaces taken with reference to an observer. This is the reason why we promote a distinction between ontological and epistemological concerns: What the agent can know about its environment depends on the agent's properties and the task. Consider the scenario of the case study: A mother is going on a flight with her 3-year-old son. In order to go from the departure hall to the gate, she first needs to check in at the check-in counter. Based on the task and the mother's properties such as being an adult, the check-in counter affords for her to put her tickets on the counter so that the check-in agent can give her the boarding passes. Although her 3-year-old son perceives the same object, namely the check-in counter, his perceived affordances are different, because of his properties, such as being too short to put something on the counter, and his task to follow his mother. This example shows how culture, experience, and intentions can highlight certain affordances rather than others (Gaver 1991).

Table 6 shows affordances perceived by an adult traveler during the task of finding her way in an airport. These (i.e., verbs) were also extracted from the interviews and synonyms merged.

Affordances belong to different realms: physical, social-institutional, and mental. *Physical affordances* require bundles of physical substance properties that match the agent's capabilities and properties—and therefore its interaction possibilities. One can only place objects on stable and horizontal surfaces, one can only drink from objects that have a brim or orifice of an appropriate size, and can be manipulated, etc. Common interaction possibilities are grasping things of a certain size with one's hands, walking on different surfaces, and moving one's eyes to perceive things. Physical affordances such as the "sittability" affordance of a chair depend on body-scaled ratios (see section 4.2.2), doorways afford going through if the agent fits through the opening, and monitors afford reflecting light depending on lighting conditions, surface properties, and the agent's viewpoint.

Substance	*Affordances*
Traveler	approach, follow, avoid, disappear, talk to, ask, provide information, behave, confirm
Check-in agent	look for, approach, talk to, ask, provide information, check in, show ticket to, behave
Stairs	go up or down, stand, wait, pay attention
Doorway	look through, enter, go through, put through
Area	look for or around, move around or through, access, leave, stand, wait, enclose, include, expect, spend time
Column	go around or towards, obstruct, block, divide
Sign	look for, go towards, stand out, recognize, check, catch one's eye, read, provide information, find one's way, advertise, follow, direct
Monitor	look for, go towards, reflect, display, search, check, read, provide information, confirm
Check-in counter	look for, go to, stand in front, line up, check in, put ticket, get boarding pass
Passport control	look for, go to or through, enter, block, line up, show passport, show boarding pass
Path	move along, branch, curve, begin or end, remember, select, direct
Decision point	look around, pass, turn, wait, decide, search, select, orient

Table 6: Affordances from substances for an adult traveler while finding her way to the gate.

Many times it is not sufficient to derive affordances from physical properties alone because people act in environments and contexts with social and institutional rules (Smith 1999). The utilization of perceived affordances, although physically possible, is often socially unacceptable or even illegal. The physical properties of passport control afford moving through. In the context of going to one's gate in an airport, passport control affords for the traveler to show her passport and boarding pass, and only then to move through. In terms of Barker this constitutes a *physical-behavioral unit* (Barker 1968), including both physical constraints and social forces. Furthermore, the whole realm of social interaction between agents is based on *social-institutional affordances*: Another traveler affords talking to, asking, and behaving in a certain way.

Physical and social-institutional affordances are the sources of *mental affordances*. In order to utilize a mental affordance, the agent needs to perform an internal operation, such as "decide." A monitor affords displaying letters and numbers, such as flight departures,

and reflecting light, but it also affords the traveler searching for her gate—i.e., performing the internal operation of matching her goal information. A path affords remembering and selecting, a decision point affords orienting and deciding, etc.

6.2 The conceptual model

Following the separation between the ontology and epistemology presented in section 6.1, our model for agent-based wayfinding simulation is two-tiered (Frank 2000). In the first tier, we consider states of the real-world environment, which are mapped to simulated environment states. In the second tier, we assume beliefs of a person about the environment. These beliefs are the result of perception and are mapped to simulated beliefs of the agent. Accordingly, percepts and actions in the real world are mapped to simulated *percepts*^{sim} and simulated *actions*^{sim} (Figure 27). The two-tiered approach allows for the integration of people's incomplete and imprecise knowledge derived from imperfect observations of space (Raubal and Worboys 1999; Worboys 1999). Furthermore, it is possible to model the perception and representation of parts, i.e., subsets, of the environment. This is important because people's knowledge of the empirical world is gained by making observations of parts of the world—resulting in subsets of affordances. A geographic space is too large and complex to allow for the observation of everything at once.

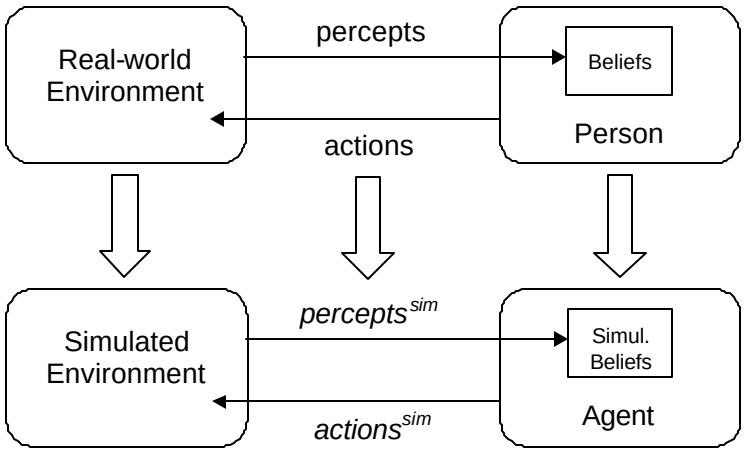

Figure 27: Mapping from real world to simulation within a two-tiered model.

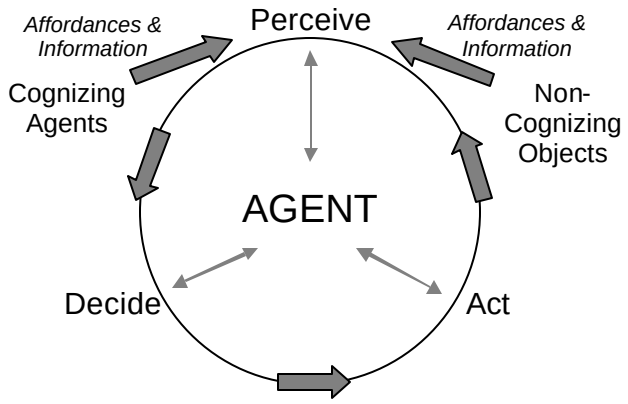

Figure 28: Conceptual process model for perceptual wayfinding.

The *perceptual wayfinding model* (Figure 28) integrates the agent's cognitive schema and perceptual structures within the *Sense-Plan-Act* framework (section 3.4). It focuses on *knowledge in the world* to explain the actions of the agent during its performance of a wayfinding task. The environment provides percepts—affordances and information from cognizing agents and non-cognizing objects—to the agent; the agent decides upon and performs actions in the environment, which in turn provides new percepts; and so on. Information such as from signs is necessary for the agent to decide upon which affordances to utilize. The internal cognitive schema guides the agent's processes of perception, decision, and action during the wayfinding task. Information about the task and goal, wayfinding strategies, and commonsense knowledge, are necessary for the agent to perform the task. The task description directs the visual perception in such a way that the agent samples only task-relevant affordances and information—therefore only a subset of all affordances and information present in the environment. The perceptual wayfinding model concentrates on the actual information needs during wayfinding and does not focus on learning the spatial environment. Its fundamental tenet is that all information must be presented at each decision point as knowledge in the world (Norman 1988).

6.2.1 Design considerations

Before designing and describing the individual components of the agent-based wayfinding simulation, we need to reconsider the questions to be answered with this tool. This helps us

Agent type	*Percepts*	*Actions*	*Goals*	*Environment*
Cognizing wayfinding agent	Affordances and information	Move	Find specific gates	Airport

Table 7: PAGE description for the cognizing wayfinding agent.

to decide which of the concepts and ideas presented so far have to be integrated and to what extent. We start with the PAGE description (see also section 4.1.2) for the cognizing wayfinding agent (Table 7).

Our main goal is to prove the hypothesis that an agent-based model for perceptual wayfinding can explain human wayfinding in an unfamiliar building. It does so by simulating the interaction of knowledge in the head and knowledge in the world. The latter is modeled through affordances and information. Furthermore, the simulation tool should answer the following questions with regard to the airport environment:

- Where do people face wayfinding difficulties?
- Why do people face these wayfinding difficulties?
- How do the wayfinding information and design have to be changed to avoid the difficulties?

Gibson's idea that the medium (section 6.1.1.1) consists of points of observation and lines of locomotion serves as the motivation to model the wayfinding environment through a graph of nodes and edges. Regarding the taxonomy of substances (section 6.1.1.2) we

Category	*Description*
Information device	In an airport the most important information regarding the task of finding one's gate comes from gate signs. Our representation of gate signs distinguishes between *single* content—e.g., "A" or "C54" (Figure 29), *list* content—e.g., "A,C" or "C52,C53" (Figure 30), and *range* content—e.g., "A–D" or "B32–B43" (Figure 31).
Gate and gate area	Represented through information such as "all gates C belong to gate area C."
Navigational element	Represented through decision points and paths.

Table 8: Categories of non-cognizing objects considered for the simulation.

represent one cognizing agent (the traveler), which has to solve a route-finding task. Non-cognizing objects are modeled on the perceptual level through affordances and information to an extent that allows for answering the before-mentioned questions. In particular, we consider the categories shown in Table 8. The surfaces (section 6.1.1.3) are necessary for people to separate the medium from the substances in the environment. They are inherent in the subject's descriptions of the space used to model the ontology and epistemology.

The most relevant physical affordance (section 6.1.2) for the wayfinding simulation is a path's affordance to move along it. We represent it explicitly in the model as the "go-to" affordance. Its utilization leads the agent from one node to another. Other physical affordances, such as a sign's affordance to reflect light, a decision point's affordance to look around, or a doorway's affordance to go through, are implicit in the model and allow for the agent's perception and locomotion. In this thesis we assume that the agent's

Figure 29: Single content.

Figure 30: List content.

Figure 31: Range content.

observations at each node are complete and also free of error with regard to the given set of affordances and information. Social-institutional affordances for the agent are inherent in the model through the semantic scope of the task in the given social setting—the physical-behavioral unit of wayfinding in an airport. Furthermore, we assume that the agent is able to utilize affordances such as to read and extract information from a sign[1]. The simulation of communication between different agents is not considered because we represent only one cognizing agent. Mental affordances are represented through the agent's decision process. Sign information affords being matched with the agent's goal information, paths

[1] What happens in people's "black boxes" during such processes has been widely discussed by cognitive scientists without leading to a common agreement—see also (Eco 1999, section 3.3.1.3 "The C(ognitive)T(ype) and the black box").

afford selection, and decision points afford searching, orienting, and deciding how to proceed. These processes are explicitly included in the model.

6.2.2 Structure of the cognizing wayfinding agent

The wayfinding agent specified in this work is defined as a *cognizing agent* (section 3.1). Such an agent can perceive affordances and information from objects and other agents in the environment, make decisions about actions according to a given task and goal, and then perform these actions. The performance of an action is the utilization of an affordance or a set of affordances.

The agent as specified in this thesis is not capable of total autonomous action (see section 4.1) because it does not have the means to learn from experience. For a fixed perceptual input, a set goal, and unchanging wayfinding strategies, the agent's behavior—its decisions and actions—is always the same and results therefore in consistent simulation outcomes. The agent is modeled as a *rational agent* that tries to maximize its performance measure—finding the way to a goal in the airport. This is done on the basis of knowledge in the world and a necessary minimum of knowledge in the head. The structure of this cognizing wayfinding agent is similar to that of a utility-based agent (section 4.1.2). Its reasoning is founded on the current state description—comprising the set of affordances and information perceived at a decision point—and the specified goal. The agent combines this with information about the results of possible actions and then chooses actions to achieve the goal. A wayfinding strategy using preferred directions serves as the utility function, allowing the cognizing agent to make a rational decision when more than one path leads from a decision point to the goal. The agent is flexible with respect to reaching different destinations: If a new destination is specified, then the agent comes up with new behavior.

The main components of the cognizing wayfinding agent are its *observation schema*, the *agent's state*, its *wayfinding strategies*, and its *commonsense knowledge* (Figure 32). The successful interaction of these components is necessary during the wayfinding task so that the agent can reach its goal. In the following sections we discuss the components in detail.

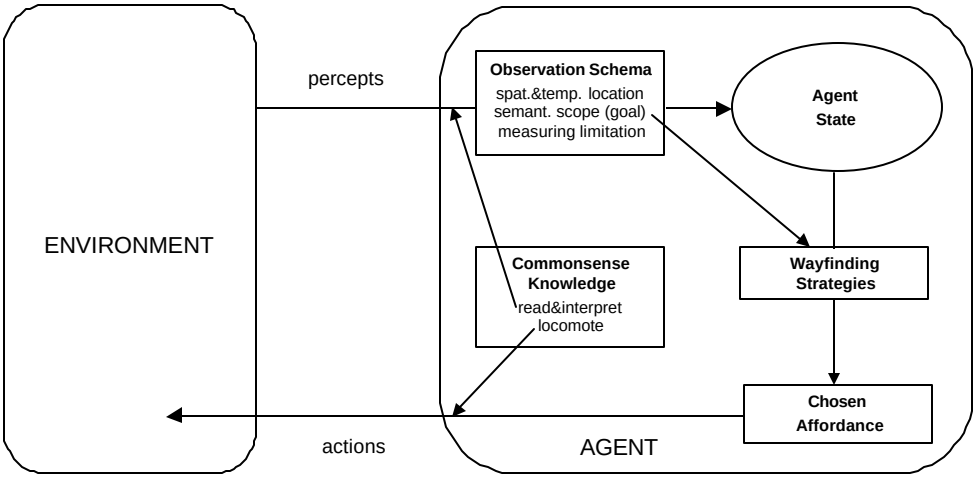

Figure 32: Interaction of components of the cognizing wayfinding agent.

6.2.2.1 Observation schema

An observation schema is the framework and context in which the agent's observations are made (Raubal and Worboys 1999; Worboys 1999). We define it based on Neisser's (1976) schema definition (section 4.4) as internal to the agent and directing what the agent perceives. The observation schema includes the spatial and temporal location at which the observations are made, the spatial and semantic scope of the observations according to the given task and goal, and possible limitations of measuring instruments. Such limitations may lead to levels of imprecision and incompleteness in the observations made with respect to it. The following example illustrates this (Raubal and Worboys 1999, p. 387):

> *An observation of a sign to a gate area A, B, or C. Due to the positioning of the sign with respect to the observer, and the style of the sign, suppose that the observer will be unable to distinguish the letters A and C. Following the observation, an observer would either gain knowledge that the sign indicates gate area A or C, or that the sign indicates gate area B. If the observation leads to knowledge that the sign indicates gate area A or C, then imprecise (and therefore certainly incomplete) knowledge has resulted.*

The observation schema directs the agent's observation of affordances and information from other cognizing agents and non-cognizing objects in the environment. This work does not focus on the process of perception itself, therefore we assume that the agent's sensors—its measuring instruments—are not limited and lead to precise and complete observation instances.

6.2.2.2 Agent state

Observations result in beliefs of the agent about some state of the environment at a specific spatial and temporal location. The cognizing wayfinding agent maintains an internal state—comparable to people's short-term memory—where all its beliefs about the environment at a specific decision point are kept until the agent moves to another decision point and represents another part of the environment (see also section 3.2.3). We model the agent's beliefs about the environment as perceived affordances and information. The knowledge available in the agent's internal state contains therefore the set of perceived affordances from which one or more are later chosen and utilized as actions—internal states of the agent are mapped to actions (see also section 4.1.1).

6.2.2.3 Wayfinding strategies

Strategies keep the agent from behaving in a random way and allow it to actually perform the wayfinding task based on a set of decision rules (section 3.2.2). The cognizing agent uses two strategies, which are hierarchically ordered (Table 9). Its *main strategy* (*Strategy 1*) is to look at each decision point for sign information containing the agent's goal and then to utilize the corresponding "go-to" affordance. This process continues until the agent has reached its goal.

Strategy	*Decision rules*	*Decision-making criteria*
Strategy 1 (main)	Does sign information contain goal information?	Sign information contains goal information.
	Does sign information match exactly with goal information?	Sign information matches exactly with goal information.
Strategy 2 (additional)	Order possible directions according to preference.	Direction with highest preference value.

Table 9: The agent uses two strategies.

At decision points where two or more possible ways lead to the goal, the agent needs to apply an *additional strategy* (*Strategy 2*). Take the following situation from the task of finding gate C57 at the Vienna International Airport (Figure 33). The agent is in front of boarding pass and ticket control and has to move through it to get closer to the goal. After moving through boarding pass and ticket control, the agent faces a decision point with three possible path continuations. Two of them—the path straight ahead to gate areas A and C, and the path to the right to gate areas B and C—are correct continuations for gate C57. Therefore the agent needs the additional strategy incorporating criteria to decide which way to go—the alternative is a random choice.

Strategy 2 is represented as a utility function in the agent's model. This function takes

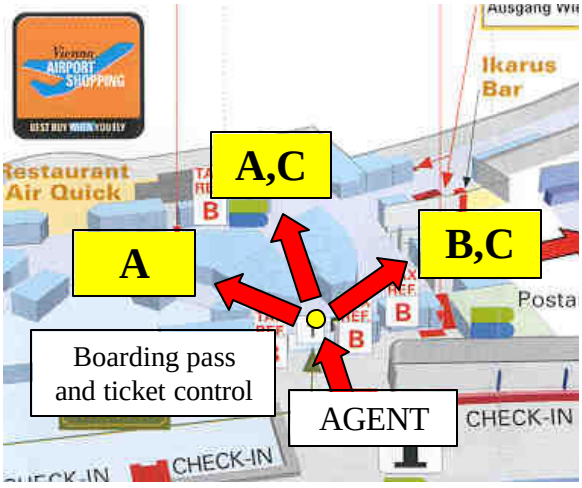

Figure 33: Wayfinding agent in front of boarding pass and ticket control at the Vienna International Airport.

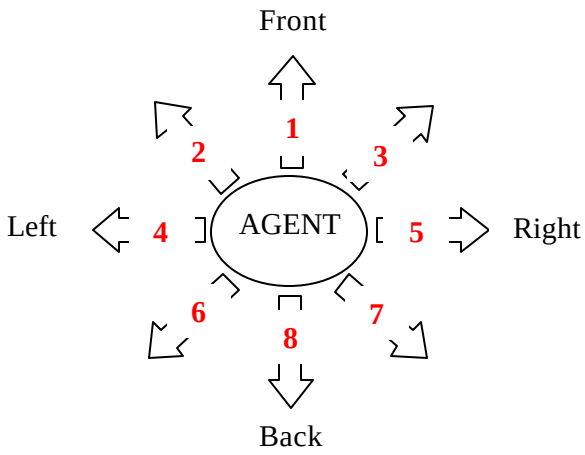

Figure 34: Directions within the agent's egocentric reference frame and their corresponding preference values.

preferred directions of the agent into account and thus allows for ordering multiple solutions to the continuation of the wayfinding process at a decision point according to associated degrees of happiness. We propose modeling preference as preferred directions within the agent's egocentric reference frame.

This reference frame is represented through eight directions—front, back, left, right, and the four directions in-between. We assume that people prefer to continue along a path in directions in front of them instead of turning around and going side- or backwards. Figure 34 shows the directions with their corresponding preference values—1 being the highest value. This wayfinding strategy is an assumption and needs to be confirmed by empirical human subjects testing. In the case of a falsification of our hypothesis, the preference values can be easily changed without influencing the other components of the agent.

6.2.2.4 Commonsense knowledge

Starting with people's first experiences with their environment they are establishing knowledge about the world in which they live. This *commonsense knowledge* is needed for everyday activities such as walking, eating, shopping, etc. It comprises many different domains that have complex interactions. Understanding a situation often involves concepts

of quantity, time, space, physics, plans, goals, needs, and communication (Davis 1990). Kuipers (1978, p. 129) defines commonsense knowledge of space as "knowledge about the physical environment that is acquired and used, generally without concentrated effort, to find and follow routes from one place to another, and to store and use the relative position of places."

This work focuses on people's information needs for reaching a goal. We do not focus on modeling all aspects of commonsense knowledge and reasoning (Lifschitz 1995) as this would go beyond the scope of the thesis. Artificial intelligence researchers have been trying for decades to formalize people's common sense and endow computer programs with it (McCarthy 1959; Hayes 1985a; Egenhofer and Mark 1995). Nevertheless, we assume that the wayfinding agent has some sort of commonsense knowledge and uses it during the simulation. Such common sense includes the abilities to

- read and understand the meaning of what is read; for example, the agent knows about the semantics of texts such as "A,B,C"—gate areas A, B, and C—and "C54"—the gate with the number 54 in gate area C; we also assume that the agent knows what symbols such as an arrow mean—i.e., follow the path in the direction of the arrow;
- perform acts of *locomotion*, defined as "coordinated behavior in response to local surrounds" by Montello (2000) who distinguishes between locomotion and wayfinding—a higher-order process involving landmarks, signs, etc.; locomotion is standing upright, avoiding barriers, or heading towards objects;

6.2.3 The simulated wayfinding environment

Wayfinding environments in the real world have a high degree of complexity (Raubal and Egenhofer 1998). They are dynamic, continuous, and most often nondeterministic (section 4.1.3). In order to represent a real-world environment in a computer system, one needs to apply mechanisms of abstraction. In this thesis we make the following three assumptions when mapping the real-world environment to the simulated environment:

1. The simulated environment is *static* and cannot change while the cognizing agent is deciding on an action. This does not have an impact on the correctness of the

simulation results because the signs and paths do not change that quickly in the real-world environment.

2. The number of possible percepts and actions for the simulated cognizing agent is limited, therefore the simulated environment is *discrete*. This is a necessary supposition to assure that the model stays computationally tractable and allows wayfinding simulations within a formal framework.

3. The cognizing agent has access to the complete, accurate, and up-to-date state of the simulated environment at every decision point—the environment is *accessible*. We make this assumption because we do not investigate wayfinding errors due to imperfect observations of space in this work.

While finding the way from one place to another in the real world, travelers consistently use sensory cues from the environment (section 3.2). Wayfinding clues, such as from signs and architectural features, are especially important at decision points—whenever a person has the opportunity to select among different paths. The number of decision points directly influences the difficulty of performing a wayfinding task (Arthur and Passini 1992; Raubal and Egenhofer 1998). Due to the importance of decision points during wayfinding we model the simulated environment through a graph, which is similar to previously used representations, such as the *wayfinding graph* presented in (Raubal and Worboys 1999), the *view graph* in (Mallot *et al.* 1999), and the *connectivity graph* in (Jiang and Claramunt 2000). *Nodes* of the graph simulate decision points in the simulated environment and have a position and state attached to them. *Edges* represent transitions between positions and states, and therefore movement of the agent between decision points. The graph has at least two distinguished nodes, the start node where the wayfinding process begins and the goal node that marks the end of the wayfinding process. We can simulate the process of wayfinding by the agent's traversal of the graph from the start state to the goal state.

6.2.4 Simulated operations

The cognizing agent performs simulated operations to make progress during wayfinding. All the simulated operations of the agent fall into one of two categories—internal operations and external operations (Table 10).

Internal operations	*External operations*	
	Perception operations	Action operations
decide	see	move
	hear	do

Table 10: Classification of operations.

Internal operations are performed on the agent's beliefs inside its mind. These beliefs are only accessible by the agent itself and they are always available to it, even without perception. Internal operations do not have an immediate effect on the environment. The agent uses them to decide what to do based on a perceived set of affordances and information—it applies its wayfinding strategies. Internal operations lead to decisions and subsequent external operations.

The performance of *external operations* directly involves the agent's environment. We further divide them into perception operations and action operations. The agent performs *perception operations* to receive input—affordances and information—from the environment. This is done by means of simulated visual perception. *Action operations* are dynamic operations in the sense that something happens in the environment—e.g., the agent moves from one decision point to the next. They do not involve information transfer to or from another cognizing agent because we do not consider communication operations such as "ask for information", "talk to another agent", etc.

The following list gives examples for the different categories of operations.

- *Decide:* choose between possible actions; resolve if goal is reached;
- *See:* observe sign, monitor; look around;
- *Move:* enter; go up / down; follow hallway; turn left / right; go straight;
- *Do:* queue up; show passport; buy goods; check in; sit; get baggage; show boarding pass;

The agent's performance of an operation corresponds to its utilization of the corresponding affordance (section 6.1.2). Physical affordances are exploited through the interaction between the agent and its environment, and therefore through the performance of external operations. Social-institutional affordances serve as constraints for the

Process model	*Real world*
Task of cognizing wayfinding agent	"Look for check-in counters 51-65"
Information from non-cognizing objects	"See signs with corresponding numbers"
Physical affordance from non-cognizing object	"Could go to counters 51-65" [path]
Internal operation	"Decide to go there"
External (action) operation = utilization of "go-to" affordance	"Go to counters 51-65"

Table 11: Process that leads to an action operation.

utilization of physical affordances and therefore external operations. Mental affordances are exploited through the agent's performance of an internal operation.

The following example taken from (Raubal 1997) demonstrates how different stages of the process model for wayfinding correspond to stages of real-world wayfinding tasks (Table 11). It shows a process that leads to an action operation.

6.3 Summary

In this chapter we developed the conceptual model for perceptual wayfinding. The model is agent-based and used to simulate people's wayfinding behavior in an unfamiliar building.

We first defined the ontology and epistemology for the agent and its environment. Both are constructed by using an ecological approach and serve as the foundations for the agent-based system. The ontology of the airport consists of a medium, substances, and surfaces. The substances are represented within both a taxonomy and a partonomy. The agent's epistemology is modeled through affordances. Affordances belong to the physical, social-institutional, or mental realm. This categorization is an extension to Gibson's theory.

Based on the ontology and epistemology we then presented the two-tiered conceptual wayfinding model. It is designed according to specific considerations that allow us to answer the posed research questions. The model consists of the cognizing agent and its wayfinding environment, which are both integrated within the Sense-Plan-Act framework from artificial intelligence. The perceptual wayfinding theory is based on the principle that all information must be presented at every decision point as knowledge in the world. In this

sense the agent perceives affordances and information at decision points, makes a decision of how to proceed, and then acts in the environment.

The agent is made of four components: its observation schema, the agent's state, its two wayfinding strategies—the main strategy and an additional one based on preference—, and commonsense knowledge. The simulated environment is an abstraction from the real world and represented through a graph structure. Regarding the simulated operations of the agent we distinguish between internal and external operations depending on their effect on the environment.

The formalization of the perceptual wayfinding model is presented in chapter 7.

CHAPTER 7

THE FORMAL MODEL FOR PERCEPTUAL WAYFINDING

This chapter presents the formalization of the conceptual model for perceptual wayfinding developed in chapter 6. The formal model is related to the conceptual model by a homomorphism (Piff 1991; Bittner and Frank 1999). For formalization we select an algebraic approach and define classes with operations in the functional language Haskell (see chapter 5). The result is an *agent-based computational model*—consisting of executable algebraic specifications—that can be used to predict people's wayfinding behavior in an unfamiliar building. We will employ it in chapter 8 to simulate various test cases from our case study.

The chapter starts with the formal representation of the cognizing agent and its wayfinding environment. We then present the formal operations of the agent within the *Sense-Plan-Act* framework. This also includes the agent's main wayfinding strategy. The formal representation of information from signs and the agent's additional wayfinding strategy are given in the subsequent sections. In the final section of this chapter we introduce the simulation framework for analyzing the agent's wayfinding process.

7.1 The cognizing agent

The cognizing wayfinding agent is formally represented as a data type, which is constructed from different types. This hierarchical structure reflects the conceptual agent model (section 6.2.2) (Figure 35).

```
data Agent = Agent AgentId ObsSchema AgentState Strategy
type AgentId = Int
data ObsSchema = ObsSchema Position Time Goal
data AgentState = AgentState [SpatialSit] PrevPosition IncomingDir Decision
data Strategy = Strategy Preferences
```

The agent has an identifier and is specified for a specific position and time instance. The latter two, together with the agent's goal, form its *observation schema*. With regard to our

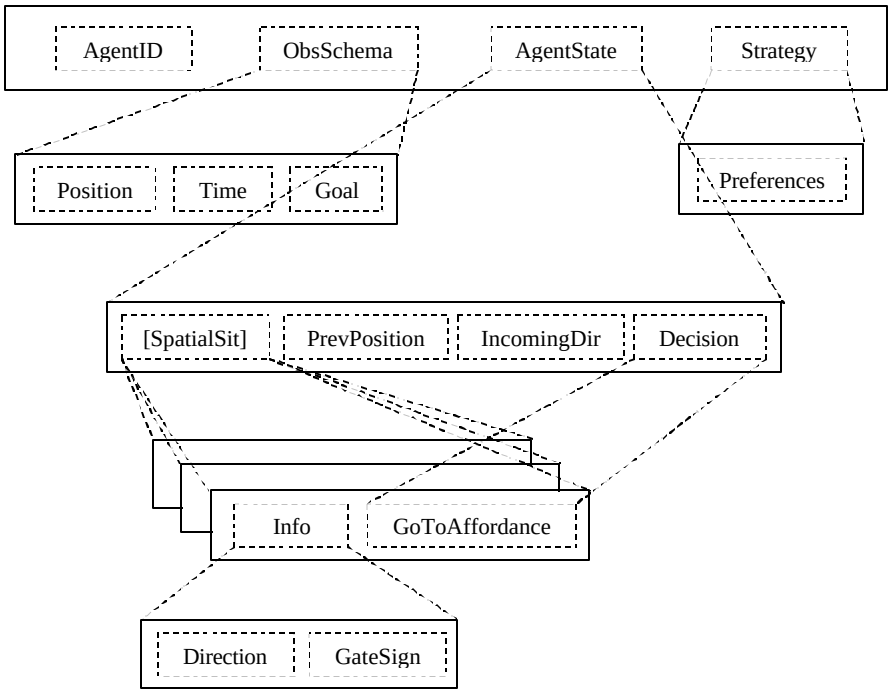

Figure 35: Hierarchical structure of data type `Agent`.

case study, the goal is defined based on the data type `Gate`—a combination of `Char` and `Int`, e.g., gate C54.

```
type Goal = Gate
data Gate = Gate Char Int
```

The state represents the agent's short-term memory and is specified through four components. First, it represents the agent's beliefs about the environment as pairs of information and "go-to" affordances. We define each individual pair as a *spatial situation*. Second, it represents the agent's previous position. Third, it represents the direction from which the agent enters a decision point. This *incoming direction* is specified within the local reference frame of the node. We need it to transform the local reference frame of a node into the egocentric reference frame of the agent. This transformation is a prerequisite for applying the agent's additional wayfinding strategy (*Strategy 2*). Finally, the state also represents the agent's decision to utilize a particular "go-to" affordance.

The data type `Strategy` is formed through a component, which is required for the agent to follow its additional wayfinding strategy. This strategy uses preferred directions of

the agent and we formally represent it as a list in which every `Direction` within the agent's egocentric reference frame is assigned a value of `Preference`.

```
type Preferences = [(Direction,Preference)]
type Direction = Int
type Preference = Int
```

7.2 The wayfinding environment

The agent's wayfinding environment is formally specified as a graph with nodes and edges, denoting decision points and transitions between the positions and states—as described in section 6.2.3. These transitions are represented as "go-to" affordances, which allow the agent to move from a node with position *x* to a connected node with position *y*.

```
data Environment = Environment Name [Node]
type Name = String
data Node = Node Position NodeState MatchDirection
```

The data type for the environment is again hierarchically structured (Figure 36). It is constructed from a name, which serves as an identifier, and a list of nodes. The nodes represent the decision points in the wayfinding environment and are specified through three components. First, every node has a position. A static environment is assumed, therefore it is not necessary to specify the point in time for each node as was done for the agent's data type (section 7.1). Second, the state of a node is defined as a list of spatial

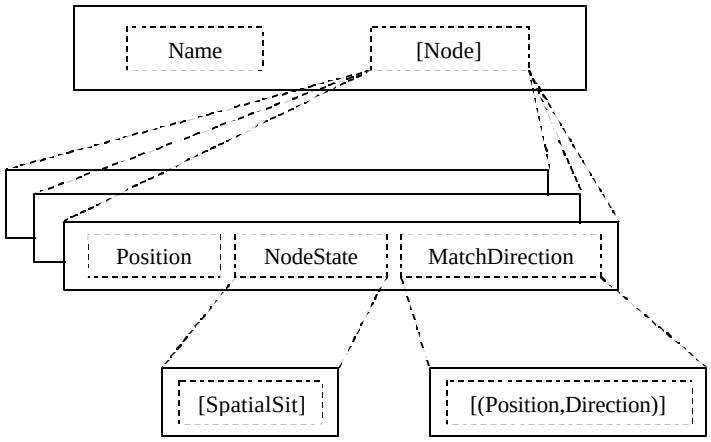

Figure 36: Hierarchical structure of data type `Environment`.

situations. Third, the type `MatchDirection` defines a table in which all the node positions from where the agent may enter a given node are assigned an incoming-direction value within the local reference frame of the given node. The direction values in each pair of the list are equal to the incoming-direction values of the agent's data type.

7.3 Formal operations of the agent

According to the classification of simulated operations in section 6.2.4, *internal and external operations* for the agent need to be formally specified. Within the *Sense-Plan-Act* (*SPA*) framework (see section 3.4) the agent senses its environment, develops a plan, and acts according to this plan. We represent these three steps through a `see` function (an external perception operation), a `decide` function (an internal operation), and an `act` function (an external action operation).

In this work the cognizing agent is specified separately from the environment. Therefore all changes, such as the change of the agent's position after a move, are represented within the agent (see section 7.1). We do not represent new states of the environment after the agent has performed an operation, because the environment is assumed to be static (see section 6.2.3). Another possible way is to simulate an environment, which contains the agent. In this case, operations of the agent lead to changes in the environment and are represented by new states of the environment after every performed operation. Formal specifications using this approach can be found in (Frank 2000) and (Bittner 2001). A philosophical discussion of modeling the *Mind* either as if contained by the *World* or put before the *World* is carried out by Eco (1999, section 1.8).

The three main functions of the *SPA*-approach are specified as operations of the class `Agents`. In addition, the function `see1` is an auxiliary function for `see`, and the function `takeStep` represents one sequence of *Sense-Plan-Act*—i.e., `see` -> `decide` -> `act`.

```
class Agents agent where
  see1 :: Node -> agent -> agent
  see :: Environment -> agent -> agent
  decide :: agent -> agent
  act :: agent -> agent
  takeStep :: Environment -> agent -> agent
```

These abstract type signatures are independent of any implementation and can therefore be implemented for different types of agents. In the following, we implement them for the data type `Agent` as defined in section 7.1, using also `Node` and `Environment` as defined in section 7.2.

7.3.1 The see function

With the `see` function we represent the agent's perception of spatial situations from the environment. The agent perceives the world at a particular point in time and its current node position. The function `getNodeAtPos` retrieves the node, whose position equals the position of the agent. This node then serves as an input parameter for the auxiliary function `see1`. Applying the function to the agent and its environment leads to changes in several components of the agent's *observation schema* and *state*:

1. The discrete time scale is incremented by one time step.
2. The new agent state comprises all spatial situations, which have been perceived by the agent at the node. A lack of any spatial situations at the node to be perceived by the agent leads to an error message including the node's position together with the agent.
3. A new incoming-direction value is assigned by looking up the corresponding value for the agent's previous position. At the first node of the wayfinding process the agent has to be given an incoming-direction value.

```
instance Agents Agent where
   see1 node agent@(Agent aid (ObsSchema p t g) (AgentState ss pp i d)
   strat)
     = (Agent aid (ObsSchema p ti g) (AgentState ssNext pp iNext d) strat)
        where
        ti = t+1
        ssNext = if (getNState node) == []
                   then error ("NO SPATIAL SITUATION AT NODE "
                    ++ showAgentPos (agent) ++ "!\n" ++ showAgent (agent))
                   else getNState node
        iNext = if pp == unit0
                   then i else unMaybe (lookup pp (getNMatchDir node))
   see env agent = see1 (getNodeAtPos (getEnvNodes env)
                   (getPosition (getObsSchema agent))) agent
```

7.3.2 The decide function

The `decide` function is an internal operation through which the agent comes to a decision of what to do next. It is designed according to the agent's two wayfinding strategies (section 6.2.2.3). Applying the function has the following effects:

1. The agent checks if it has already reached its goal (*Strategy 1*). If yes, then the wayfinding task is completed and the agent is shown. If not, then the following changes in the agent's time and decision will occur.
2. The discrete time scale is incremented by one time step.
3. The agent is looking for a match between its goal information and the perceived sign information—which is now in its short-term memory. A negative result of this matching process leads to an error message including the node's position together with the agent. If there is a match, then the corresponding "go-to" affordance becomes the result of the agent's decision-making process (*Strategy 1*). With more than one piece of gate-sign information matching the goal, the agent chooses an affordance according to its preference (*Strategy 2*). This process, using the functions `nodeToPref` and `sortSpatialSits` is explained in section 7.5.

```
instance Agents Agent where
   decide agent@(Agent aid (ObsSchema p t g) (AgentState ss pp i d) strat)
     = if isAtGoal agent
        then error ("REACHED GOAL " ++ showAgent (agent))
        else (Agent aid (ObsSchema p ti g) (AgentState ss pp i dNext) strat)
          where
          ti = t+1
          dNext = if ss == []
                  then unit0
                  else if (filter ((matchGateSign(g)) . getInfoSign
                                   . getInfo) ss) == []
                       then error ("NO MATCHING SIGN INFORMATION AT NODE "
                                   ++ showAgentPos (agent) ++ "!\n"
                                   ++ showAgent (agent))
                       else (getAff (head (sortSpatialSits (nodeToPref
                         agent (filter ((matchGateSign(g)) . getInfoSign
                         . getInfo) ss)))))
```

7.3.3 The act function

The `act` function simulates the agent's utilization of an affordance. Affordances are possibilities for behavior, therefore it is necessary to distinguish between the agent's decision to utilize a particular "go-to" affordance—the result of the `decide` function—and the agent's performance of the action offered by the "go-to" affordance—the result of the `act` function. Applying the function to the agent leads to changes in several components of the agent's *observation schema* and *state*.

1. The discrete time scale is incremented by one time step.
2. The agent's position gets a new value, which equals the end position of the "go-to" affordance.
3. The list of spatial situations in the agent's state becomes empty because the agent has moved to a new node and has not perceived any of the new spatial situations yet.
4. The previous position gets a new value.

5. No decision has been made at the new node therefore the decision component is empty.

```
instance Agents Agent where
  act (Agent aid (ObsSchema p t g) (AgentState ss pp i d) strat)
    = (Agent aid (ObsSchema pNext ti g) (AgentState ssNext ppNext i dNext)
        strat) where
                ti = t+1
                (pNext,ssNext) = if d==unit0
                                 then (p,ss)
                                 else (getEnd(d),[])
                ppNext = p
                dNext = unit0
```

One sequence of *Sense-Plan-Act* specifies the process of the agent moving from one node to another. We implement this sequence by composing the functions `see`, `decide`, and `act` together into the new function `takeStep`.

```
instance Agents Agent where
  takeStep env agent = (act . decide . (see env)) agent
```

7.4 Formal representation of information from signs

Wayfinding information is represented by the data type `Info`. Its constructor function takes a direction and a gate sign as input parameters. The direction defines the bearing of the sign's corresponding path—given through the spatial situation it belongs to—within the local reference frame of the node.

```
data SpatialSit = SpatialSit Info GoToAffordance
data Info = Info Direction GateSign
```

The distinction between three types of gate signs based on a sign's information content (section 6.2.1) is specified through the data type `GateSign`. The elements of its constructor functions are the data types `GateSignSingle`, `GateSignList`, and `GateSignRange`. *Single content* can be either a letter defining a gate area, or a combination of letter and number defining an individual gate. In addition, the data type `AtGate` is specified to indicate a gate sign marking the end node of a wayfinding task.

List content represents a list of gate areas or a list of individual gates. *Range content* can be either a range between two gate areas or a range between two individual gates.

```
data GateSign = GateSign GateSignSingle | GateSign1 GateSignList |
   GateSign2 GateSignRange
data GateSignSingle = GateSignSingle LetterOnly | GateSignSingle1 Gate |
   GateSignSingle2 AtGate
data GateSignList = GateSignList [LetterOnly] | GateSignList1 [Gate]
data GateSignRange = GateSignRange LetterOnly LetterOnly |
   GateSignRange1 Gate Gate
```

We can now define the function `matchGateSign`, which is executed when the agent pursues *Strategy 1* (section 7.3.2). It allows the agent to evaluate whether its goal information matches with any of the perceived information from signs. This means that based on its goal information (e.g., C54) the agent can decide if a piece of sign information is relevant for reaching the goal (e.g., C) or not (e.g., A). Instances of the function `matchGateSign` are specified for the type signature in the class `GateSigns`—for different types of information content. As examples we show the two instances representing a match between the goal and sign information with list content. The other instances can be found in the Appendix.

```
class GateSigns gateSign where
   matchGateSign :: Goal -> gateSign -> Bool
instance GateSigns GateSign where
   matchGateSign (Gate goal_l goal_n) (GateSign1 (GateSignList onlyLetters))
        = elem goal_l (map getLetterOnlyLetter onlyLetters)
   matchGateSign goal (GateSign1 (GateSignList1 gates))
        = elem goal gates
```

7.5 Formal representation of the agent's additional wayfinding strategy

The agent's additional wayfinding strategy (*Strategy 2*, see section 6.2.2.3) of having preferred directions when more than one path leads from a decision point to the goal, is specified through three stages—starting with the list of all spatial situations with sign information matching the agent's goal information.

1. The directions of sign information within the local reference frame of the node are transformed into directions within the egocentric reference frame of the agent at the same node (`nodeDirsToAgentDirs`). This transformation is based on the function `nodeDirToAgentDir`, which takes as inputs an information direction and the agent's incoming direction, and produces as a result another information direction within the agent's reference frame.
2. The directions within the agent's egocentric reference frame are transformed into the preferred directions of the agent (`agentDirsToPrefs`).
3. The preferred directions of the agent are sorted according to the order of preference (`sortSpatialSits`).

The functions used to represent *Strategy 2* are specified in the classes `Agents`, `IncomingDirs`, and `SpatialSits`. For better readability, steps 1 and 2 are combined to the function `nodeToPref` using function composition.

The function `orderSpatialSits` sorts two spatial situations according to their preference values and is used within the function `sortSpatialSits`—based on `sortLs` (Thompson 1999, p. 189). Both of their instances are given in the Appendix.

```
class Agents agent where
  nodeDirsToAgentDirs :: agent -> [SpatialSit] -> [SpatialSit]
  agentDirsToPrefs :: agent -> [SpatialSit] -> [SpatialSit]
  nodeToPref :: agent -> [SpatialSit] -> [SpatialSit]
instance Agents Agent where
  nodeDirsToAgentDirs agent@(Agent aid obs state strat) spatialSits
    = map nodeDirToAgentDir' spatialSits
        where nodeDirToAgentDir' (SpatialSit (Info dir gateSign) aff)
          = SpatialSit (Info (nodeDirToAgentDir dir (getIncomingDir state))
              gateSign) aff
  agentDirsToPrefs agent@(Agent aid obs state strat) spatialSits
    = map lookupPref spatialSits
        where lookupPref (SpatialSit (Info dir gateSign) aff)
          = SpatialSit (Info (unMaybe (lookup dir (getPreferences strat)))
              gateSign) aff
  nodeToPref agent = agentDirsToPrefs agent . nodeDirsToAgentDirs agent
```

```
class IncomingDirs incomingDir where
  nodeDirToAgentDir :: Direction -> incomingDir -> Direction
instance IncomingDirs IncomingDir where
  nodeDirToAgentDir dir incDir = mod (dir + (4 - incDir)) 8
class SpatialSits spatialSit where
  orderSpatialSits :: spatialSit -> spatialSit -> Bool
  sortSpatialSits  :: [spatialSit] -> [spatialSit]
```

The agent uses the result of this process for deciding on a "go-to" affordance within the `decide` function (see section 7.3.2).

7.6 Formal analysis of the wayfinding simulation

The formal specifications developed in this chapter need to be integrated to build the functions for analyzing the agent's wayfinding process in an unfamiliar building. The core `simulation` function is specified in the class `Agents`.

```
class Agents agent where
  simulation :: Environment -> agent -> IO()
instance Agents Agent where
  simulation env agent = output where
    output = putStrLn ((showTitle env agent)
      ++ concat (map showAgent (wayfinding env agent))
      ++ (showCycle (findCycle env agent)) ++ showCheckNodes
      ++ concat (map showNode (map (getNodeAtPos (getEnvNodes env))
      (init (findCycle env agent)))))
```

It takes an environment and agent as inputs and produces different outputs depending on the outcome of the wayfinding process (Figure 37). The following four results are possible:

1. The agent has reached its goal—stop of simulation is caused by the `decide` function (section 7.3.2.).
2. The simulation halts because the agent cannot perceive any spatial situations at a node and therefore does not know what to do. This is caused by the `see` function (section 7.3.1).

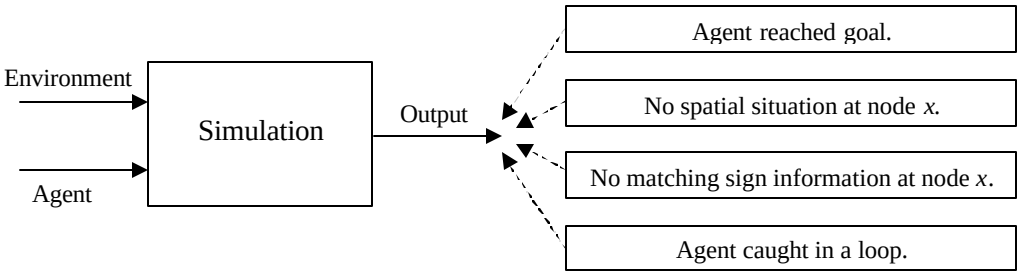

Figure 37: Input and output for the `simulation` function.

3. The simulation halts because the agent cannot find any sign information that matches with its goal information. Therefore it does not know how to proceed further. The halt is caused by the `decide` function (section 7.3.2).
4. The agent has been caught in a loop because the sign information containing the goal is pointing the wrong way at a node.

The major subfunctions for the simulation—`wayfinding` and `findCycle`—are explained below.

7.6.1 The wayfinding function

Simulating the agent's wayfinding behavior and analyzing such behavior is done by creating an infinite list of data types `Agent`—the same agent but for each position during the wayfinding task. This is possible because of Haskell's lazy evaluation strategy (section 5.3.2.4).

```
class Agents agent where
  wayfinding :: Environment -> agent -> [agent]
instance Agents Agent where
  wayfinding env agent = take (complexity env) (iterate (takeStep env)
       agent)
```

The `wayfinding` function terminates the computation of list elements after a certain number of steps. This is necessary because otherwise an agent caught in a loop would lead to infinite computation. The number of elements after which the list gets truncated depends on the *complexity* of the environment. The complexity is defined as the longest possible

path from any start node to any goal node and depends therefore on the number of decision points in the environment (Raubal and Egenhofer 1998). The function `complexity` is specified in the class `Environments`. It derives the number of undirected edges in the graph by extracting from the environment all "go-to" affordances and dropping the duplicates. For example, the two "go-to" affordances `(3,4)` and `(4,3)` belong to the same undirected edge.

```
class Environments environment where
   complexity :: environment -> Int
instance Environments Environment where
   complexity env = length (dropDuplicateEdges (nodesToAffs (getEnvNodes
        env)))
```

7.6.2 The findCycle function

The `findCycle` function is used to find the loop—a *cycle* in graph theory (section 5.1)—the agent is caught in. It starts from a list of all consecutive positions the agent has reached during the wayfinding process—i.e., the output of the function `positionsOfAgent`. From this list the cycle gets extracted.

```
class Agents agent where
   positionsOfAgent :: Environment -> agent -> [Position]
   findCycle :: Environment -> agent -> [Position]
instance Agents Agent where
   positionsOfAgent env agent = skeletOfList (map getPosition (map
        getObsSchema (wayfinding env agent)))
   findCycle env agent = findCycle1 (findRepeatingCycle
        (positionsOfAgent env agent))
```

The specifications for the auxiliary functions `skeletOfList`, `findCycle1`, and `findRepeatingCycle` are given in the Appendix.

7.7 Summary

This chapter presented the formalization of the agent-based model for perceptual wayfinding. We employed an algebraic approach and used the functional programming language Haskell—defining classes with operations. This resulted in a set of *executable*

specifications, which will be used in the next chapter to simulate various wayfinding tasks at the Vienna International Airport.

We started by specifying the data types for the cognizing agent and its wayfinding environment according to their conceptual models. Both of these types consist of different components, which are embedded in a hierarchical structure.

Based on the *Sense-Plan-Act* framework and the agent's wayfinding strategies we then defined the formal operations of the agent. These are the *see*-, *decide*-, and *act* functions. Taken together, they represent one sequence of Sense-Plan-Act. The functions are implemented for the agent- and environment data types as specified before.

Wayfinding information from signs was formally represented by taking into account our case study wayfinding in an airport. The specifications reflect therefore the distinction between the three different types of gate signs based on a sign's information content. Furthermore, we specified a function simulating the agent's process of matching its goal information with the perceived sign information.

The agent's additional wayfinding strategy was represented as a three-stage-process. It is followed by the agent when more than one path leads from a decision point to the goal. This strategy is based on preferred directions.

In the final section of this chapter we designed the simulation framework for analyzing the agent's wayfinding process in an unfamiliar building. This was done by integrating the formal specifications constructed earlier. The simulation takes two inputs—the agent and its environment—and can produce four different outputs depending on the wayfinding process.

CHAPTER 8

AGENT-BASED SIMULATION OF WAYFINDING AT THE VIENNA INTERNATIONAL AIRPORT

In chapters 6 and 7 we developed and formally specified the perceptual model for agent-based wayfinding simulation. This chapter puts the formal specifications to a test by applying them to our case study. Our goal is not only to test the validity of the model by checking whether the simulation of different test cases yields plausible results, but also to answer our research questions and verify the hypothesis of this thesis.

The case study is wayfinding at the Vienna International Airport (chapter 2). The agent has to find its way from one of the check-in counters to a specific gate. We are interested in whether the agent is able to reach its goal based on the sign information and "go-to" affordances offered at different decision points, and if not, where and why the agent faces wayfinding difficulties, and what can be done to avoid them. We will demonstrate three different outcomes of the simulation. In the first case the agent reaches its goal. In the second case the agent cannot reach its goal because a node lacks the matching sign information. Finally, we show a case where the agent is caught in a loop.

The simulation takes inputs and produces outputs (section 7.6). The chapter starts with a description of both the test data for the wayfinding environment and the specification of the agent. We then present for each case the agent's task and the outcome of the simulation. The chapter concludes with an assessment of the simulation results.

8.1 The test data

The test data consist of a graph representation for the departure level of the Vienna International Airport (VIE) and an instance for the cognizing agent at the start of each test case.

8.1.1 Test data for the wayfinding environment

The source of these test data are pictures of decision points, hallways, and signs taken by the author at VIE (see chapter 2). The wayfinding environment is represented by a directed graph with 45 nodes (Figure 38). The directed edges stand for the "go-to" affordances, which in most cases have sign information connected to them. Sometimes though, signs are missing or lack any information. As an example, Table 12 shows the test data for the nodes 3, 4, 5, and 6—their positions, the "go-to" affordances with the connected sign information and bearings, and also the incoming-direction values assigned to adjacent nodes from where the agent could have entered. The complete test data are given in the Appendix.

Position	*Go-to*	*Sign*	*Direction*	*(Enter from, incoming dir.)*
3	5	A,C	0	(2,4),(4,1),(5,0),(6,6)
	4	A	1	
	6	B,C	6	
4	32	A	2	(3,5),(5,6),(31,3),(32,2)
	31	-	3	
	3	-	5	
	5	B,C	6	
5	4	A	2	(3,4),(4,2),(7,6)
	3	-	4	
	7	B	6	
	7	C	6	
6	7	C	0	(3,3),(7,0),(26,6)
	3	-	3	
	26	B	6	

Table 12: Node states for the test environment.

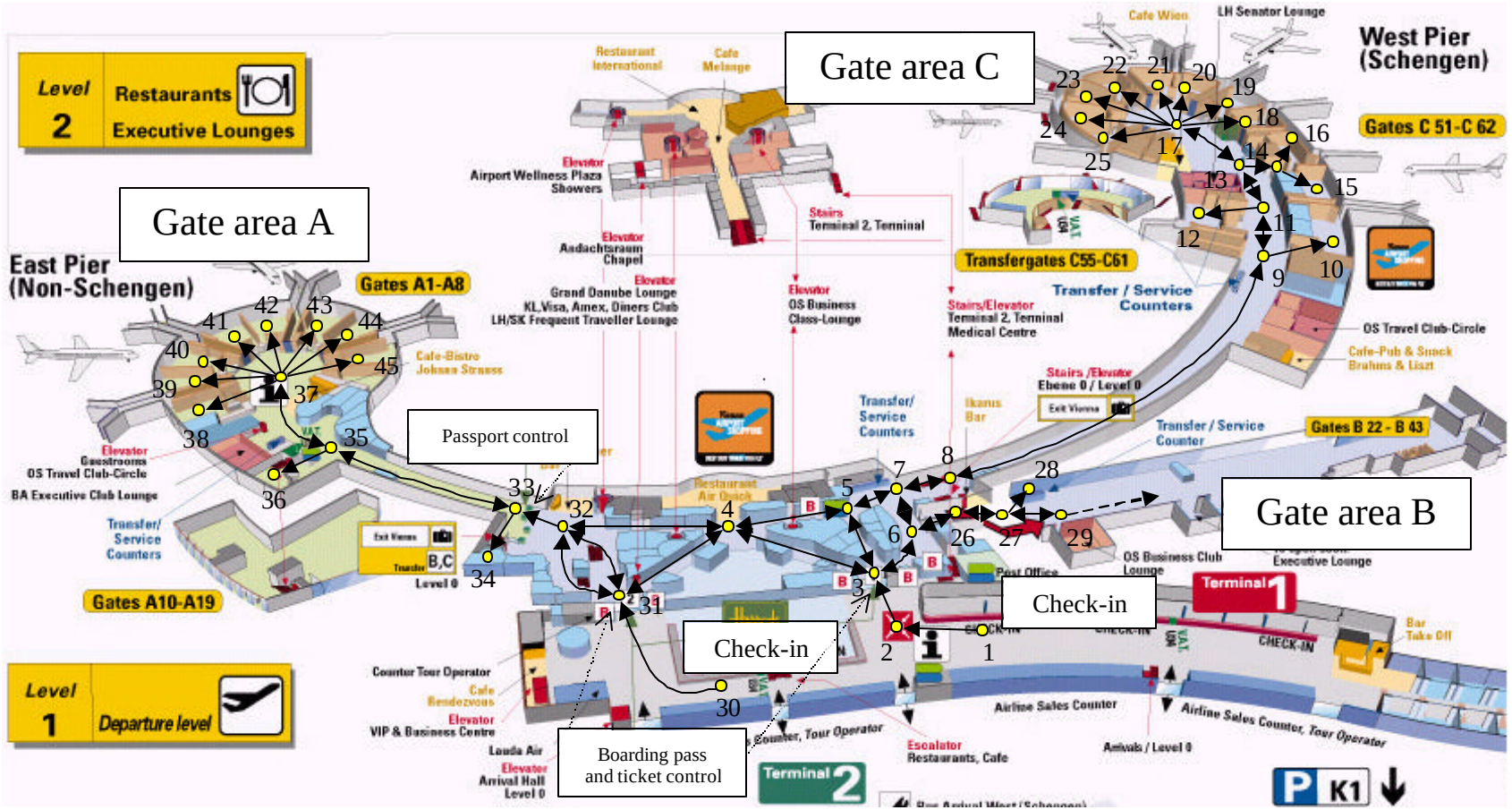

Figure 38: Representation of the wayfinding environment.

The formal specification of the environment appears as follows. In the case of a lack of a sign or sign information the `GateSign` is assigned `unit0`.

```
vie = Environment "Vienna Int. Airport" [node1,…,node45]
node3 = Node 3 [SpatialSit (Info 0 (GateSign1 (GateSignList [(LetterOnly
   'A'),(LetterOnly 'C')]))) (3,5), SpatialSit (Info 1 (GateSign
   (GateSignSingle (LetterOnly 'A')))) (3,4), SpatialSit (Info 6 (GateSign1
   (GateSignList [(LetterOnly 'B'),(LetterOnly 'C')]))) (3,6)]
   [(2,4),(4,1),(5,0),(6,6)]
node4 = Node 4 [SpatialSit (Info 2 (GateSign (GateSignSingle (LetterOnly
   'A')))) (4,32), SpatialSit (Info 3 unit0) (4,31), SpatialSit (Info 5
   unit0) (4,3), SpatialSit (Info 6 (GateSign1 (GateSignList [(LetterOnly
   'B'),(LetterOnly 'C')]))) (4,5)] [(3,5),(5,6),(31,3),(32,2)]
node5 = Node 5 [SpatialSit (Info 2 ( GateSign (GateSignSingle (LetterOnly
   'A')))) (5,4), SpatialSit (Info 4 unit0) (5,3), SpatialSit (Info 6
   (GateSign (GateSignSingle (LetterOnly 'B')))) (5,7), SpatialSit (Info 6
   (GateSign (GateSignSingle (LetterOnly 'C')))) (5,7)] [(3,4),(4,2),(7,6)]
node6 = Node 6 [SpatialSit (Info 0 (GateSign (GateSignSingle (LetterOnly
   'C')))) (6,7), SpatialSit (Info 3 unit0) (6,3), SpatialSit (Info 6
   (GateSign (GateSignSingle (LetterOnly 'B')))) (6,26)]
   [(3,3),(7,0),(26,6)]
```

One can allocate the orientation of the local reference frames to the nodes of the wayfinding environment in an arbitrary way. It makes sense though to adjust them in such a way that the number of axes pointing exactly to the bearings of “go-to” affordances is a maximum. This facilitates the process of assigning directions to the sign information connected to the outgoing paths. Figure 39 shows the local reference frame for node 3, which is the decision point after boarding pass and ticket control. The signs “A,C”, “A”, and “B,C” are localized at the directions 0, 1, and 6.

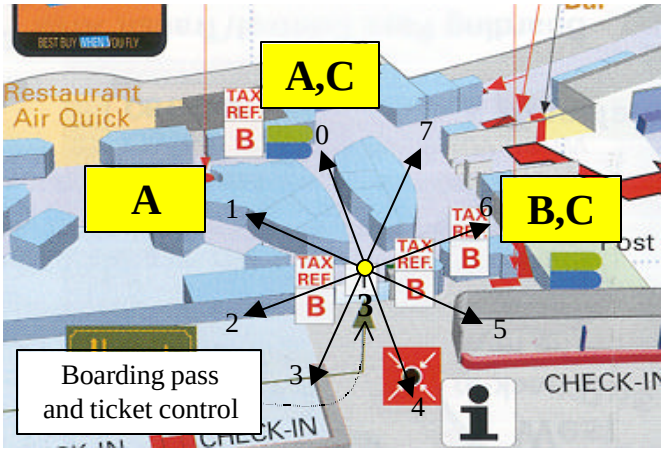

Figure 39: Local reference frame for node 3—the decision point after boarding pass and ticket control.

8.1.2 Test data for the cognizing agent

The cognizing agent needs an instance at the beginning of the simulation. We employ the same instance for all three test cases, except that a different identifier, position, and goal are used for each of them. The following instance of the agent is at the start of its wayfinding task for test case 1.

```
agent1 = Agent 1 (ObsSchema 1 1 (Gate 'A' 6)) (AgentState [] unit0 7 unit0)
  (Strategy pref)
pref = [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]
```

Agent *1* begins at the position *1* and point in time *1*. Its goal is to find *gate A6*. The list of spatial situations within the agent's state is empty because the agent has not perceived anything yet. The agent's previous position is assigned *zero* (`unit0`) and an arbitrary value for the incoming direction is specified. The agent has not made any decision therefore this value is also *zero* (`unit0`). The preferred directions of the agent are specified according to Figure 40. The numbers the arrows point to stand for the cardinal directions North (*0*), North-West (*1*), West (*2*), South-West (*3*), South (*4*), South-East (*5*), East (*6*), and North-East (*7*). These are the directions within the agent's egocentric reference frame, therefore they correspond to front, back, left, right, etc. (Frank 1996). The ranking for the preferred directions is given as the numbers from *1* (highest preference) to

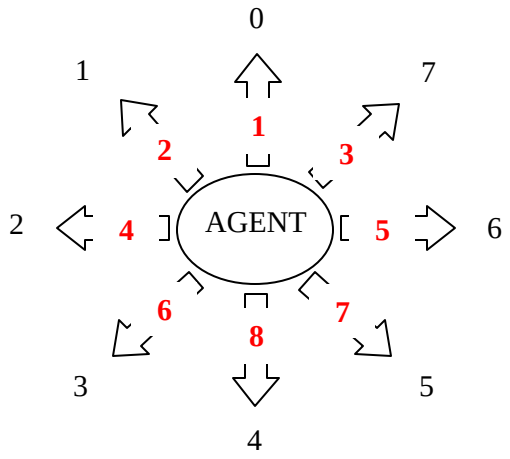

Figure 40: Preferred directions of the agent.

8 (lowest preference) inside the arrows. The preferred directions are represented through `pref`.

8.2 Test case 1: Agent reaches goal

The first test case demonstrates a scenario where the agent can reach its goal by utilizing knowledge in the world offered to it at decision points.

8.2.1 The agent's task

The agent's task is to find the way from a check-in counter in Terminal 1 (node 1) to gate A6 (node 43) located in the East Pier of the airport. The agent's representation at the beginning of the simulation was given in section 8.1.2.

8.2.2 The outcome of the simulation

The output of the simulation is a list of agent representations—one after every step the agent has performed during the wayfinding task (section 7.6.1). In the following we present the full representation of the agent after its first step and then only those components of the agent that have changed. The complete output is given in the Appendix.

```
AGENT 1
   OBSERVATION SCHEMA: Pos. = 1, Time = 1, Goal = 'A'6
   AGENT STATE: Spatial Sits. = none; Prev. Pos. = 0, Inc. Dir. = 7,
        Decision = none
   STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]
   OBSERVATION SCHEMA: Pos. = 2, Time = 4
   AGENT STATE: Prev. Pos. = 1, Inc. Dir. = 7
   OBSERVATION SCHEMA: Pos. = 3, Time = 7
   AGENT STATE: Prev. Pos. = 2, Inc. Dir. = 6
   OBSERVATION SCHEMA: Pos. = 5, Time = 10
   AGENT STATE: Prev. Pos. = 3, Inc. Dir. = 4
   OBSERVATION SCHEMA: Pos. = 4, Time = 13
   AGENT STATE: Prev. Pos. = 5, Inc. Dir. = 4
   OBSERVATION SCHEMA: Pos. = 32, Time = 16
   AGENT STATE: Prev. Pos. = 4, Inc. Dir. = 6
   OBSERVATION SCHEMA: Pos. = 33, Time = 19
   AGENT STATE: Prev. Pos. = 32, Inc. Dir. = 6
   OBSERVATION SCHEMA: Pos. = 35, Time = 22
   AGENT STATE: Prev. Pos. = 33, Inc. Dir. = 6
   OBSERVATION SCHEMA: Pos. = 37, Time = 25
   AGENT STATE: Prev. Pos. = 35, Inc. Dir. = 5
   OBSERVATION SCHEMA: Pos. = 43, Time = 28
   AGENT STATE: Prev. Pos. = 37, Inc. Dir. = 4
REACHED GOAL
   OBSERVATION SCHEMA: Pos. = 43, Time = 29
   AGENT STATE: Spatial Sits. = (Info: 0 - at gate 'A'6, GoTo: none);
        Prev. Pos. = 37, Inc. Dir. = 3
```

The result shows that the agent has reached its goal at the point in time 29. Its path led from the start node 1 via the nodes 2, 3, 5, 4, 32, 33, 35, and 37 to the end node 43. Note that at node 3 the agent used its additional wayfinding strategy of preferred directions and moved straight ahead (see also Figure 39), therefore taking a longer path. This situation

illustrates the argument that people cannot apply criteria such as shortest path in an unfamiliar environment because they do not have access to information about what lies ahead of them (section 3.2.2).

8.3 Test case 2: Agent cannot match goal with sign information

The second test case simulates a scenario where the agent cannot proceed further at a node because it has not perceived any sign information, which matches its goal information.

8.3.1 The agent's task

The agent's task is to find the way from a check-in counter in Terminal 2 (node 30) to gate C56 (node 20) located in the West Pier of the airport. The representation of the agent at the start of the simulation is as follows.

```
agent2 = Agent 2 (ObsSchema 30 1 (Gate 'C' 56)) (AgentState [] unit0 7
  unit0) (Strategy pref)
```

8.3.2 The outcome of the simulation

The following list gives the agent's full representation after its first step and then only those components that have changed.

```
AGENT 2
  OBSERVATION SCHEMA: Pos. = 30, Time = 1, Goal = 'C'56
  AGENT STATE: Spatial Sits. = none; Prev. Pos. = 0, Inc. Dir. = 7,
       Decision = none
  STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

  OBSERVATION SCHEMA: Pos. = 31, Time = 4
  AGENT STATE: Prev. Pos. = 30, Inc. Dir. = 7

  OBSERVATION SCHEMA: Pos. = 4, Time = 7
  AGENT STATE: Prev. Pos. = 31, Inc. Dir. = 4

  OBSERVATION SCHEMA: Pos. = 5, Time = 10
  AGENT STATE: Prev. Pos. = 4, Inc. Dir. = 3

  OBSERVATION SCHEMA: Pos. = 7, Time = 13
```

```
   AGENT STATE: Prev. Pos. = 5, Inc. Dir. = 2

   OBSERVATION SCHEMA: Pos. = 8, Time = 16
   AGENT STATE: Prev. Pos. = 7, Inc. Dir. = 2

   OBSERVATION SCHEMA: Pos. = 9, Time = 19
   AGENT STATE: Prev. Pos. = 8, Inc. Dir. = 3

   OBSERVATION SCHEMA: Pos. = 11, Time = 22
   AGENT STATE: Prev. Pos. = 9, Inc. Dir. = 3

NO MATCHING SIGN INFORMATION AT NODE 11!
   OBSERVATION SCHEMA: Pos. = 11, Time = 23
   AGENT STATE: Spatial Sits. = (Info: 0 - none, GoTo: 11->13)
        (Info: 2 – 'C'62, GoTo: 11->12) (Info: 4 - ["'A'","'B'"],
        GoTo: 11->9); Prev. Pos. = 9, Inc. Dir. = 4, Decision = none
```

The agent correctly traveled a path leading to gate area C passing the nodes 30, 31, 4, 5, 7, 8, and 9. At node 11 the agent stopped because it did not perceive any sign information matching its goal information. The agent's state at node 11 shows that the agent perceived a "go-to" affordance offering a continuation of the path to node 13 and then further on to the goal node 20 (see Figure 38). But this affordance does not have any sign information connected to it (Figure 41). Passengers in the real world master this situation by utilizing the "go-to" affordance for node 13 and therefore go straight ahead,

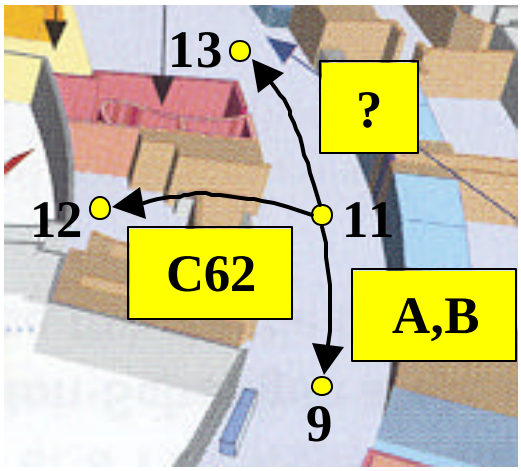

Figure 41: "Go-to" affordances and sign information at node 11.

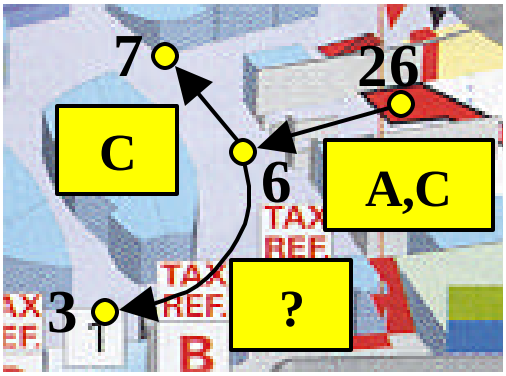

Figure 42: Missing sign information for gate area A at node 6.

because they exclude the two other possibilities—turning left for gate C62 or going back to where they came from. Nevertheless, the result of the simulation is useful because it points out a consistency problem to the designer. Node 11 is the only decision point on this path, which lacks sign information regarding the specified goal. For some passengers this might result in a feeling of uncertainty. Therefore a sign with information on where the path straight ahead leads should be added.

We found a similar but more critical situation for the task of transferring from gate area B to gate area A. At node 26 the agent perceives sign information indicating a path to gate area A. At the next decision point (node 6) there are two paths to continue—one leading to node 3 and the other one leading to node 7—but there is no sign information for gate area A anymore (Figure 42). This is a serious conundrum and was also reiterated by an airport official working at the nearby boarding pass and ticket control. He complained that many passengers ask him every day for the way to gate area A. The simulation output for this scenario is given below.

```
AGENT 3
   OBSERVATION SCHEMA: Pos. = 6, Time = 4, Goal = 'A'6
   AGENT STATE: Prev. Pos. = 26, Inc. Dir. = 7
NO MATCHING SIGN INFORMATION AT NODE 6!
   OBSERVATION SCHEMA: Pos. = 6, Time = 5
   AGENT STATE: Spatial Sits. = (Info: 0 - 'C', GoTo: 6->7)
        (Info: 3 - none, GoTo: 6->3) (Info: 6 - 'B', GoTo: 6->26);
        Prev. Pos. = 26, Inc. Dir. = 6, Decision = none
```

8.4 Test case 3: Agent is caught in a loop

The third test case shows a scenario where the agent does not reach its goal because it is caught in a loop.

8.4.1 The agent's task

When simulating wayfinding between check-in counters and different gates with the given test data we found that the agent never gets caught in a loop. Therefore the sign information at node 4 is manipulated to demonstrate such a result. This is done by exchanging the gate signs for the directions 2 and 5. The formal specification of node 4 is now:

```
node4 = Node 4 [SpatialSit (Info 2 unit0) (4,32), SpatialSit (Info 3 unit0)
  (4,31), SpatialSit (Info 5 (GateSign (GateSignSingle (LetterOnly 'A'))))
  (4,3), SpatialSit (Info 6 (GateSign1 (GateSignList [(LetterOnly
  'B'),(LetterOnly 'C')]))) (4,5)] [(3,5),(5,6),(31,3),(32,2)]
```

The agent's task for test case 3 is to find the way from a check-in counter in Terminal 1 (node 1) to gate A6. The representation of the agent at the start of the simulation is as follows.

```
agent4 = Agent 4 (ObsSchema 1 1 (Gate 'A' 6)) (AgentState [] unit0 7 unit0)
  (Strategy pref)
```

8.4.2 The outcome of the simulation

We give again the agent's full representation after its first step and then only those components, which have changed.

```
AGENT 4
  OBSERVATION SCHEMA: Pos. = 1, Time = 1, Goal = 'A'6
  AGENT STATE: Spatial Sits. = none; Prev. Pos. = 0, Inc. Dir. = 7,
       Decision = none
  STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

  OBSERVATION SCHEMA: Pos. = 2, Time = 4
  AGENT STATE: Prev. Pos. = 1, Inc. Dir. = 7

  OBSERVATION SCHEMA: Pos. = 3, Time = 7
```

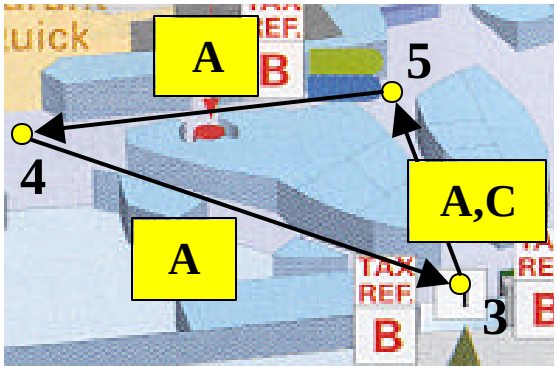

Figure 43: Fictive cycle caused by misinformation at node 4.

```
AGENT STATE: Prev. Pos. = 2, Inc. Dir. = 6

...

OBSERVATION SCHEMA: Pos. = 5, Time = 136
AGENT STATE: Prev. Pos. = 3, Inc. Dir. = 1

OBSERVATION SCHEMA: Pos. = 4, Time = 139
AGENT STATE: Prev. Pos. = 5, Inc. Dir. = 4

OBSERVATION SCHEMA: Pos. = 3, Time = 142
AGENT STATE: Prev. Pos. = 4, Inc. Dir. = 6

Agent has been caught in a loop: [3,5,4,3]
Sign information for goal pointing the wrong way at one of these nodes!
```

The result shows that the agent could not reach its goal because it was caught in a cycle consisting of nodes 3, 5, and 4. This was caused by the fact that the agent was misinformed at node 4 where it was directed back to node 3 (Figure 43). Looking at the test data for node 4 one can see that the reason for this misinformation is that the sign with the information on how to get to gate area A points in a wrong direction. To avoid this wayfinding problem for passengers the direction of the sign has to be changed.

8.5 Assessment of the simulation

Based on the outcomes of the simulation we assess that the proposed computational theory for perceptual wayfinding gives plausible results with respect to wayfinding at the Vienna

International Airport. The test cases demonstrate that *the agent-based model for perceptual wayfinding is a valid model to simulate people's wayfinding behavior in this environment* and can be used as a tool to point out wayfinding problems caused by misinformation.

- The first test case shows that the agent reaches its goal when it does not encounter any wayfinding problems caused by misinformation or missing information. It also demonstrates that the proposed additional wayfinding strategy works correctly with regard to its specification in the conceptual and formal model.
- The second test case points out a problem spot where the sign information for the continuing path to the goal is missing. Although not a major problem for passengers in the airport this might cause a feeling of uncertainty in some of them and shows a consistency problem to the designer. A further example demonstrates another problem spot where the sign information for the way to a gate area is missing. This is a major conundrum, which has been often encountered by passengers in the real environment.
- The third test case is a fictitious example but nevertheless gives a plausible result. It points out a loop based on misinformation at a decision point to the designer.

The simulation shows where and why the agent faces wayfinding problems in the airport. The case of wayfinding in an unfamiliar airport is a representative example for wayfinding in an unfamiliar building, which typically works with the help of sign information leading the ways to different goals; see, for example, public transport terminals, hospitals, university buildings, and libraries. We therefore conclude that *human wayfinding in unfamiliar buildings can be explained on the basis of the agent-based model for perceptual wayfinding developed in this thesis.*

The simulation demonstrates how the agent finds the way to a gate in the airport by utilizing "go-to" affordances and their connected information. Such *affordance-information pairs*—more specifically, a "go-to" affordance offering a path to the goal and its connected information correctly indicating that this path leads to the goal—must be presented to the wayfinder at every decision point. This is *the minimum amount of knowledge in the world, which is necessary for the agent to find a specific goal.*

Although the focus of the perceptual wayfinding model is on knowledge in the world represented in the form of "go-to" affordances and information, a minimum of knowledge

in the head is required to perform the wayfinding tasks. *The agent's observation schema, state, wayfinding strategies, and commonsense knowledge form this minimum set of internal components*:

1. The *observation schema* includes the spatial and temporal context and the semantic scope of the task. Without a specified goal the agent would move aimlessly in the environment.
2. The *agent's state* serves as short-term memory for its perceptions and decisions. The short-term representation of the perceived knowledge comprises the set of all the "go-to" affordances and information perceived at a decision point and is the basis for the agent's decision-making process. We distinguish between decision and action, therefore the agent needs to keep its decision in mind—at least until it gets utilized as an action.
3. The *wayfinding strategies* are necessary for the agent to make decisions. Rational behavior would be impossible without them.
4. The agent needs some sort of *commonsense knowledge*, otherwise it could neither read and interpret the information on a sign, nor locomote between decision points.

We therefore conclude that *the agent-based model for perceptual wayfinding simulates the interaction of a minimum amount of knowledge in the head and knowledge in the world*.

8.6 Summary

In this chapter we tested the perceptual model for agent-based wayfinding by simulating different test cases from our case study. The agent had to find the ways from check-in counters to gates at the Vienna International Airport.

We first gave a description of the test data. The wayfinding environment is specified by a directed graph. Its nodes represent the decision points in the airport and its edges stand for the "go-to" affordances, which are connected to sign information. The cognizing agent is specified for the start of each wayfinding task.

We then simulated three test cases representing three different wayfinding scenarios. In the first case the agent does not encounter any wayfinding problems and reaches its

goal. In the second case the agent does not reach its goal because a decision point lacks sign information matching the agent's goal information. In the third case, which is a fictitious example, the agent cannot reach its goal because it is caught in a loop.

In the final section of this chapter the simulation results of the test case were assessed. We come to the conclusion that the agent-based model for perceptual wayfinding yields plausible results and can be used to explain people's wayfinding behavior in unfamiliar buildings. It simulates the interaction of a minimum of knowledge in the head and knowledge in the world.

CHAPTER 9

CONCLUSIONS AND FUTURE WORK

This chapter begins with a summary of the research done in this thesis. It describes all the stages we went through for developing, formalizing, and testing the agent-based model for perceptual wayfinding. We then present the results and major findings of our work. Finally, we propose various directions for future research.

9.1 Summary

The goal of this thesis was to develop a *computational theory of perceptual wayfinding*. This theory uses an agent-based approach and can explain people's wayfinding behavior in unfamiliar buildings. The agent-based model focuses on *knowledge in the world* but also includes *knowledge in the head*. We set out to find the minimum set of the agent's knowledge in the head and the minimum amount of knowledge in the world necessary to allow for the agent's navigation.

Wayfinding at the Vienna International Airport was used as a case study. This particular domain serves as a representative example for wayfinding in an unfamiliar building. It also allowed us to reduce the complexity of human wayfinding to a manageable level. In chapter 2 we introduced the particulars of wayfinding in an airport in general and described the setting and task of our case study in particular.

Developing a theory of perceptual wayfinding is an interdisciplinary endeavor. As a starting point we used theories and concepts from different scientific fields. Chapter 3 explained previous work on modeling human wayfinding—from spatial cognition and research on how people find their ways, to mental representations. We also looked at computational wayfinding models and work done in artificial intelligence. These theories were linked to our agent-based model. In chapter 4 we introduced the main modeling concepts employed to develop the theory of perceptual wayfinding. Agent theory is employed as a conceptual paradigm for the perceptual wayfinding model. An ecological

scientific viewpoint is taken for representing the agent's structures of perception and cognition. We integrated Gibson's theory of affordances by extending it with elements of cognition, situational aspects, and social constraints. The formal methods to construct the computational model were explained in chapter 5. A graph is used to represent the wayfinding environment. The formal agent-based model consists of algebraic specifications, which are based on the mathematical concept of algebra. In this work, the functional programming language Haskell was chosen to express these specifications.

Chapter 6 described the conceptual model for perceptual wayfinding. In order to build a spatial process model that describes itself as being a simulator of human behavior, this model must be grounded in people's experiences. We therefore used empirical data from human subjects testing to build the ontology and epistemology for the agent-based model. Both were derived by employing an ecological approach, that is, by focusing on the information transactions between living systems and their environments. The resulting ontology consists of a medium, substances, and surfaces, and depicts what is in the airport. The substances were thereby represented within both a taxonomy and a partonomy. The epistemology is modeled through affordances and describes the agent's knowledge and beliefs.

The conceptual model is based on the ontology and epistemology. It therefore consists of two tiers: simulated states of the environment and simulated beliefs of the agent. The model was developed according to specific design considerations, which allowed us to answer the research questions posed at the beginning of this thesis. The individual components of the model were designed to have minimal functionality for achieving the set objective—i.e., the agent's ability to find a goal in the airport based on knowledge in the world. The perceptual wayfinding theory is based on the principle that all information must be presented to the wayfinder at every decision point as knowledge in the world.

The perceptual wayfinding model integrates the agent and its environment within a *Sense-Plan-Act* framework. The components of the cognizing wayfinding agent are:

- its *observation schema* defining the framework and context of the agent's observations,
- its *state* representing the agent's beliefs about the environment,
- two *wayfinding strategies* necessary for the agent to make rational decisions, and

- *commonsense knowledge* allowing the agent to locomote and understand the meaning of signs;

The wayfinding environment is modeled as a graph, where nodes simulate different decision points and edges simulate lines of movement.

When performing a wayfinding task, the agent starts with a goal description at a start node. During the navigation process it accumulates beliefs about the environment by observing task-relevant "go-to" affordances and their connected sign information at decision points. Affordances are possibilities for action with reference to the agent. Information is necessary for the agent to decide upon which affordances to utilize. The utilization of a "go-to" affordance leads the agent from one node to another where it is again provided with new percepts. A successful navigation corresponds to the agent's traversal from a start to a goal node.

In chapter 7 the conceptual model for perceptual wayfinding was formalized as an algebra within the functional programming environment Haskell. The resulting executable specifications formally describe the interaction between the agent and its environment, and fix the meaning of the conceptual model. We defined the data types for the agent and the environment, which are both embedded in a hierarchical structure. According to the Sense-Plan-Act framework, the three functions *see*, *decide*, and *act* were formally specified and implemented for the agent and environment data types. We further specified the agent's wayfinding strategies and wayfinding information from signs according to our case study. Finally, the simulation framework for analyzing the agent's wayfinding process in the airport was specified by integrating these specifications.

In chapter 8 the agent-based model for perceptual wayfinding was tested by simulating the task of finding the way from a check-in counter to specific gates at the Vienna International Airport. The simulation comprises three scenarios—with each of them having a different outcome. The results demonstrated that the proposed algebraic specifications of the agent-based wayfinding simulation developed in this thesis allow us to analyze the wayfinding process of a cognizing agent in an unfamiliar building. It is possible to determine whether the agent is able to reach its goal based on knowledge in the world, and if not, where and why wayfinding problems occur and what needs to be done to avoid them. We finally concluded that the agent-based model for perceptual wayfinding developed in this thesis can explain people's wayfinding behavior in an unfamiliar

building. This model simulates the interaction of a minimum amount of knowledge in the head and knowledge in the world.

9.2 Results and major findings

The major scientific result of this thesis is the *formal agent-based wayfinding model*. This model is *based on the theory of perceptual wayfinding* and *explains people's wayfinding behavior in an unfamiliar building.* It is different from previous computational models for wayfinding, which were built to investigate how mental representations are created, stored, and used. These models assume that people become familiar with their environments over time and therefore acquire cognitive maps. In many situations though, people have to find their ways to novel destinations in unfamiliar environments—such as finding their gate in an airport they have never been to before. Our model concentrates on people's actual *information needs* during wayfinding and does not focus on learning a spatial environment. Its main principle is that all wayfinding information about the different destinations has to be presented to the wayfinder at every decision point as *knowledge in the world*. Such knowledge needs to be perceived by the wayfinder and we therefore call our model the *model for perceptual wayfinding*.

From an engineering point of view, the main result of this work is a *practical tool* that can be used *to test the wayfinding information* presented to people in an environment. The agent-based simulation framework allows analysis of the agent's wayfinding process with respect to success or failure of reaching a goal. It discovers where and why wayfinding problems for the agent occur and what needs to be done to avoid them. This tool can help designers and architects to test and assess possible design alternatives prior to the construction of a building.

The formal agent-based model consists of *algebraic specifications written in the functional programming language Haskell*. Algebraic specifications allow for describing the structure of abstract data types and their operations. For that reason they are particularly suited for the representation of change. The agent-based paradigm focuses on the activities of the agent: An agent can perceive its environment and act in this environment. This leads to various changes—both in the agent and its environment. Algebraic specifications are therefore an appropriate and useful method to formalize agents and their behavior.

The thesis further demonstrates the *minimum amount of knowledge in the world and knowledge in the head necessary for the agent to find its goal*. The minimum amount of knowledge in the world is described by *affordance-information pairs*—also called *spatial situations*. In particular, at every decision point there must be a "go-to" affordance offering a path to the goal and its connected information correctly indicating that this path leads to the goal. The components of the cognizing wayfinding agent form a minimum set because all of them are necessary to achieve cognitively plausible results. Without the *observation schema* the agent would be missing the context and goal of the task and therefore not know what to observe from the complex environment. The agent's *state* is required to model beliefs about the world. These are combined with the goal information to decide upon possible task-relevant actions. Without the help of the *wayfinding strategies* the agent would not be able to make rational decisions. Finally, without any form of *commonsense knowledge* the agent would be unable to locomote between decision points, make sense of wayfinding information, or know what is meant by different symbols such as an arrow on a sign.

Our model is grounded in people's spatial experiences and therefore conceptually sound and cognitively plausible (Raubal 1997). We described an *ecological approach to model the ontology and epistemology for agent-based wayfinding simulation in the airport domain*. The ontology of the airport environment is based on a subdivision into medium, substances, and surfaces. The epistemological model uses the concept of affordances, which we divide into *physical*, *social-institutional*, and *mental affordances*. The idea that affordances belong to different realms is new because Gibson did not distinguish between different classes of affordances. Such a distinction is useful because it allows the imposition of constraints on the utilization of physical affordances through social and institutional rules—and therefore also through cultural conventions. Spatial situations, which offer physical and social-institutional affordances, often create mental affordances for an agent—e.g., the agent needs to decide which of the affordances to utilize.

9.3 Directions for future work

In this thesis we made various simplifying assumptions for the design of the agent-based model for perceptual wayfinding. Only those concepts and processes absolutely necessary to answer our research questions were taken into consideration. The model could be

extended by explicitly integrating all elements of the proposed ontology and other physical and social-institutional affordances from the epistemology. This would on the one hand allow for the testing of various other tasks in the airport, such as finding the closest emergency exit or finding the way to the baggage claim area. On the other hand it would make it possible to simulate subtasks, such as checking in at the check-in counter, going to the restrooms, or buying goods at a duty-free store.

The construction of the ontology and epistemology presented here is based on interviews with human subjects concerning wayfinding in airports. More testing, such as performed by Mark *et al.* (1999b), needs to be done to see if proper categories were formed and to test different instances for category membership. Future research in this area has to include different behavioral environments—ontological and epistemological theories have to be developed and integrated to extend the agent-based simulation tool. Further work might focus on the influence of attributes such as color on the perception of affordances.

People's knowledge of the empirical world results from their perception of parts of the world. This knowledge is usually incomplete and imprecise. The two-tiered structure of our agent-based model for perceptual wayfinding allows for the integration of wayfinding errors such as encoding errors due to poor perceptual recording or recognition errors (Golledge 1999). Considering these errors and integrating a filter mechanism that selects the most relevant affordances and information for a given task would improve the model to simulate human behavior more closely.

It is not clear how people visually and semantically connect different affordances with individual pieces of information. How does a person know that a piece of information is related to one affordance rather than to another? In the model presented here, we represented related affordance-information pairs as spatial situations based on human subjects testing but the agent cannot do this automatically. Future work by psychologists and cognitive scientists might provide the answers to this question.

In this thesis we specified one agent that is able to find a goal in a specific environment. In the real world people also have the possibility to communicate with other people—for example, ask another person when they get lost. The simulation of social interaction between agents requires including multiple agents and communication operations within a multi-agent system. Bittner (2001) specified such a multi-agent system for the domain of cadastre. His work demonstrated the importance of developing a

practical ontology for the design process of information systems. In general, the representation of communication between multiple agents is still a difficult problem due to the lack of such domain-specific ontologies (Nwana and Ndumu 1999).

In order to assess the results of the simulation applied to the case study, one needs to compare them to the results of human subjects testing in the real environment. Such a comparison will also help to test various parameters of the model, such as the proposed wayfinding strategies for the agent, and find additional ones to be included. For example, previous tests with human subjects have shown that emotions are important for making rational decisions (Picard 1997; Trappl *et al.* forthcoming). People who are intelligent but do not have any emotional capacities face problems while making rational decisions because their search space is not restricted. Empirical data can be used to refine the model components so that the simulation results match real-world processes more closely. The importance of field-testing to develop human-like characterizations for artificial agents has already been pointed out by Gimblett *et al.* (1997) among others.

Larger test cases need to be carried out to see whether Haskell serves as a testing tool that is efficient with regard to computational performance and cost. If this is not the case, then the proposed specifications need to be implemented using a different tool. The specifications might also be implemented within a robot, which finds its way in the real world or a physical model of it. This would help to detect major problems, such as the before mentioned connection between affordances and pieces of information, more easily.

BIBLIOGRAPHY

G. Allen (1999) Spatial Abilities, Cognitive Maps, and Wayfinding - Bases for Individual Differences in Spatial Cognition and Behavior. in: R. Golledge (Ed.), *Wayfinding Behavior - Cognitive Mapping and Other Spatial Processes*. pp. 46-80, Johns Hopkins University Press, Baltimore.

P. Arthur and R. Passini (1990) *1-2-3 Evaluation and Design Guide to Wayfinding*. Public Works Canada , Technical Report.

P. Arthur and R. Passini (1992) *Wayfinding: People, Signs, and Architecture.* McGraw-Hill Ryerson, Toronto.

R. Barker (1968) *Ecological Psychology - Concepts and Methods for Studying the Environment of Human Behavior.* Stanford University Press, Stanford.

G. Bingham and M. Muchisky (1995) "Center of Mass Perception": Affordances as Dispositions by Dynamics. in: J. Flack, P. Hancock, J. Caird, and K. Vicente (Eds.), *Global Perspectives on the Ecology of Human-Machine Systems*. 1, pp. 359-395, Lawrence Erlbaum Associates, Hillsdale, New Jersey.

R. Bird and P. Wadler (1988) *Introduction to Functional Programming*. Prentice Hall International, Hemel Hempstead (UK).

S. Bittner (2001) *An agent-based model of reality in a cadastre*. Ph.D. thesis, Technical University Vienna, Vienna, Austria.

T. Bittner and A. Frank (1999) On the design of formal theories of geographic space. *Journal of Geographical Systems* 1(3): 237-275.

R. Brooks (1991a) Intelligence Without Reason. in: *IJCAI'91, 12th Int. Joint Conference on Artificial Intelligence*, Sydney, Australia, pp. 569-595.

R. Brooks (1991b) Intelligence without representation. *Artificial Intelligence* (47): 139-159.

R. Brooks, C. Breazeal, M. Marjanovic, B. Scassellati, and M. Williamson (1998) The Cog Project: Building a Humanoid Robot. in: C. Nehaniv (Ed.), *Computation for Metaphors, Analogy and Agents*. *Lecture Notes in Artificial Intelligence* 1562, Springer.

P. Brown (2001) A Ray of Hope for Air Travelers Following Signs. The New York Times, July 10th, 2001. http://www.nytimes.com/2001/06/07/living/07AIR.html?ex=992930267&ei=1&en=8ba5175f067b5df0.

J. Bryson (2000) Cross-Paradigm Analysis of Autonomous Agent Architecture. *Journal of Experimental and Theoretical Artificial Intelligence* 12(2): 165-190.

A. Car (1996) *Hierarchical Spatial Reasoning: Theoretical Consideration and its Application to Modeling Wayfinding*. Department of Geoinformation, Technical University Vienna, Vienna.

A. Car and A. Frank (1995) Formalization of Conceptual Models for GIS using Gofer. *Computers, Environment, and Urban Systems* 19(2).

B. Chandrasekaran, J. Josephson, and R. Benjamins (1999) What Are Ontologies, and Why Do We Need Them? *IEEE Intelligent Systems* (1): 20-26.

A. Cohn (1995) The Challenge of Qualitative Spatial Reasoning. *ACM Computing Surveys* 27(3): 323-325.

E. Cornell, D. Heth, and D. Alberts (1994) Place recognition and way finding by children and adults. *Memory & Cognition* 22(6): 633-643.

E. Davis (1990) *Representations of Commonsense Knowledge*. Morgan Kaufmann Publishers, Palo Alto.

R. Downs and D. Stea (1977) *Maps in Minds: Reflections on Cognitive Mapping*. Harper and Row, New York.

U. Eco (1999) *Kant and the Platypus - Essays on Language and Cognition*. Harvest Book Harcourt, San Diego, New York.

M. Egenhofer and D. Mark (1995) Naive Geography. in: A. Frank and W. Kuhn (Eds.), *Spatial Information Theory - A Theoretical Basis for GIS. Lecture Notes in Computer Science* 988, pp. 1-15, Springer, Berlin-Heidelberg-New York.

S. Epstein (1997) Spatial Representation for Pragmatic Navigation. in: S. Hirtle and A. Frank (Eds.), *Spatial Information Theory—A Theoretical Basis for GIS, International Conference COSIT '97, Laurel Highlands, PA. Lecture Notes in Computer Science* 1329, pp. 373-388, Springer, Berlin.

Flughafen-Wien-AG (2001a) 28.06.01: IATA Passenger Survey - Top Ratings for Vienna International Airport. http://english.viennaairport.com/pr_ausg.cfm?ID=50.

Flughafen-Wien-AG (2001b) Vienna International Airport - Departure. http://english.viennaairport.com/abflug.html.

A. Frank (1992) Spatial Reasoning—Theoretical Considerations and Practical Applications. in: J. Harts, H. Ottens, H. Scholten, and D. Ondaatje (Eds.), *EGIS'92, Third European Conference and Exhibition on Geographical Information Systems*, Munich, Germany, pp. 310-319.

A. Frank (1996) Qualitative spatial reasoning: cardinal directions as an example. *International Journal of Geographical Information Systems* 10(3): 269-290.

A. Frank (1999) One Step up the Abstraction Ladder: Combining Algebras - From Functional Pieces to a Whole. in: C. Freksa and D. Mark (Eds.), *Spatial Information Theory - Cognitive and Computational Foundations of Geographic Information Science, International Conference COSIT '99*, Stade, Germany, pp. 95-107.

A. Frank (2000) Spatial Communication with Maps: Defining the Correctness of Maps Using a Multi-Agent Simulation. in: C. Freksa, W. Brauer, C. Habel, and K. Wender (Eds.), *Spatial Cognition II - Integrating Abstract Theories, Empirical Studies, Formal Methods, and Practical Applications. Lecture Notes in Artificial Intelligence* 1849, pp. 80-99, Springer, Berlin, Heidelberg, Germany.

A. Frank (forthcoming-a) Pragmatic Information Content - How to Measure the Information in a Route Description.

A. Frank (forthcoming-b) Ontology for Spatio-Temporal Databases. in: T. Sellis (Ed.), *Spatiotemporal Databases: The Chorochronos Approach*. Springer.

A. Frank and W. Kuhn (1995) Specifying Open GIS with Functional Languages. in: M. Egenhofer and J. Herring (Eds.), *Advances in Spatial Databases (SSD'95). Lecture Notes in Computer Science* 951, pp. 184-195, Springer, Portland, ME, USA.

A. Frank and W. Kuhn (1999) A Specification Language For Interoperable GIS. in: M. Goodchild, M. Egenhofer, R. Fegeas, and C. Kottman (Eds.), *Interoperating Geographic Information Systems. The Kluwer International Series in Engineering and Computer Science* pp. 123-132, Kluwer Academic Publishers, Boston / Dordrecht / London.

A. Frank, M. Raubal, and M. van der Vlugt, Eds. (2000) *PANEL-GI Compendium - A guide to GI and GIS*. Geographical Information Systems International Group (GISIG) & European Commission, Genova, Italy.

C. Freksa (1991) Qualitative Spatial Reasoning. in: D. Mark and A. Frank (Eds.), *Cognitive and Linguistic Aspects of Geographic Space. NATO Advanced Science Institutes Series D: Behavioural and Social Sciences - Vol. 63* pp. 361-372, Kluwer Academic Press, Dordrecht, Boston, London.

C. Freksa (1992) Using Orientation Information for Qualitative Spatial Reasoning. in: A. Frank, I. Campari, and U. Formentini (Eds.), *Theories and Methods of Spatio-Temporal Reasoning in Geographic Space. Lecture Notes in Computer Science* 639, pp. 162-178, Springer.

T. Gärling, A. Böök, and E. Lindberg (1986) Spatial orientation and wayfinding in the designed environment: A conceptual analysis and some suggestions for postoccupancy evaluation. *Journal of Architectural Planning Resources* 3: 55-64.

T. Gärling, E. Lindberg, and T. Mäntylä (1983) Orientation in buildings: Effects of familiarity, visual access, and orientation aids. *Journal of Applied Psychology* 68: 177-186.

E. Gat (1998) On Three-Layer Architectures. in: D. Kortenkamp, P. Bonnasso, and R. Murphy (Eds.), *Artificial Intelligence and Mobile Robots*. AAAI Press.

W. Gaver (1991) Technology Affordances. in: *Human Factors in Computing Systems, CHI'91 Conference Proceedings*. pp. 79-84, ACM Press.

J. Gibson (1977) The Theory of Affordances. in: R. Shaw and J. Bransford (Eds.), *Perceiving, Acting, and Knowing - Toward an Ecological Psychology*. pp. 67-82, Lawrence Erlbaum Ass., Hillsdale, New Jersey.

J. Gibson (1979) *The Ecological Approach to Visual Perception*. Houghton Mifflin Company, Boston.

D. Gilbert, M. Aparicio, B. Atkinson, S. Brady, J. Ciccarino, B. Grosof, P. O'Connor, D. Osisek, S. Pritko, R. Spagna, and L. Wilson (1995) *The Role of Intelligent Agents in the Information Infrastructure.* IBM Corporation, Research Triangle Park, NC, USA, Technical Report.

H. Gimblett, D. Durnota, and R. Itami (1997) Some Practical Issues in Designing and Calibrating Artificial Human-Recreator Agents in GIS-based Simulated Worlds. *Complex International Journal - Workshop on Comparing Reactive (ALife-ish) and Intentional Agents* 3.

M. Gluck (1991) Making Sense of Human Wayfinding: Review of Cognitive and Linguistic Knowledge for Personal Navigation with a New Research Direction. in: D. Mark and A. Frank (Eds.), *Cognitive and Linguistic Aspects of Geographic Space*. *Series D: Behavioural and Social Sciences* 63, pp. 117-135, Kluwer Academic Publishers, Dordrecht, The Netherlands.

R. Golledge (1992) Place Recognition and Wayfinding: Making Sense of Space. *Geoforum* 23(2): 199-214.

R. Golledge (1995) Path Selection and Route Preference in Human Navigation: A Progress Report. in: A. Frank and W. Kuhn (Eds.), *Spatial Information Theory-A Theoretical Basis for GIS*. *Lecture Notes in Computer Science* 988, pp. 207-222, Springer, Berlin-Heidelberg-New York.

R. Golledge (1999) Human Wayfinding and Cognitive Maps. in: R. Golledge (Ed.), *Wayfinding Behavior - Cognitive Mapping and Other Spatial Processes*. pp. 5-45, Johns Hopkins University Press, Baltimore.

R. Golledge and R. Stimson (1997) *Spatial Behavior: A Geographic Perspective*. Guilford Press, New York.

S. Gopal, R. Klatzky, and T. Smith (1989) NAVIGATOR: A Psychologically Based Model of Environmental Learning Through Navigation. *Journal of Environmental Psychology* (9): 309 - 331.

S. Gopal and T. Smith (1990) Human way-finding in an urban environment: a performance analysis of a computational process model. *Environment and Planning A* 22: 169 - 191.

J. Guttag, E. Horowitz, and D. Musser (1978) The Design of Data Type Specifications. in: R. Yeh (Ed.), *Current Trends in Programming Methodology*. Vol. 4 - Data Structuring, pp. 60-79, Prentice-Hall, Englewood Cliffs, NJ.

P. Hayes (1985a) The Second Naive Physics Manifesto. in: J. Hobbs and B. Moore (Eds.), *Formal Theories of the Commonsense World*. pp. 1-36, Ablex, Norwood, NJ.

P. Hayes (1985b) Naive Physics I: Ontology For Liquids. in: J. Moore (Ed.), *Theories of the Commonsense World*. pp. 71-89, Ablex Publishing Corporation.

H. Heft (1996) The Ecological Approach to Navigation: A Gibsonian Perspective. in: J. Portugali (Ed.), *The Construction of Cognitive Maps*. *The GeoJournal Library* 32, pp. 105-132, Kluwer Academic Publishers, Dordrecht/Boston/London.

S. Hirtle (1998) The Cognitive Atlas: Using GIS as a Metaphor for Memory. in: M. Egenhofer and R. Golledge (Eds.), *Spatial and Temporal Reasoning in Geographic Information Systems*. pp. 267-276, Oxford University Press.

P. Hudak, J. Peterson, and J. Fasel (2000) A Gentle Introduction to Haskell - Version 98. http://www.haskell.org/tutorial/.

G. Janzen, T. Herrmann, S. Katz, and K. Schweizer (2000) Oblique Angled Intersections and Barriers: Navigating through a Virtual Maze. in: C. Freksa, W. Brauer, C. Habel, and K. Wender (Eds.), *Spatial Cognition II - Integrating Abstract Theories, Empirical Studies, Formal Methods, and Practical Applications*. *Lecture Notes in Artificial Intelligence* 1849, pp. 277-294, Springer, Berlin, Heidelberg, Germany.

B. Jiang and C. Claramunt (2000) Extending Space Syntax Towards an Alternative Model of Space Within GIS. in: *3rd AGILE Conference*, Helsinki, Finland.

P. Kirschenhofer (1995) The Mathematical Foundation of Graphs and Topology for GIS. in: A. Frank (Ed.), *Geographic Information Systems - Materials for a Post-Graduate Course, Vol. 1: Spatial Information*. *GeoInfo Series* 4, pp. 155-176, Department of Geoinformation, TU Vienna, Vienna.

R. Kitchin (1994) Cognitive Maps: What are they and why study them? *Journal of Environmental Psychology* 14: 1-19.

K. Koffka (1935) *Principles of Gestalt Psychology*. Harcourt Brace, New York.

W. Kuhn (1996a) Handling Data Spatially: Spatializing User Interfaces. in: M. Kraak and M. Molenaar (Eds.), *SDH'96, Advances in GIS Research II, Proceedings*. 2, pp. 13B.1-13B.23, International Geographical Union, Delft.

W. Kuhn (1996b) *Semantics of Geographic Information*. Department of Geoinformation, Vienna.

W. Kuhn (2000) Ontologies from Texts. in: A. Caschetta (Ed.), *GIScience 2000*, Savannah, Georgia, USA, pp. 49-51.

B. Kuipers (1978) Modeling Spatial Knowledge. *Cognitive Science* (2): 129-153.

B. Kuipers (1982) The 'Map in the Head' Metaphor. *Environment and Behaviour* 14(2): 202-220.

B. Kuipers, R. Froom, W.-Y. Lee, and D. Pierce (1993) The Semantic Hierarchy in Robot Learning. in: J. Mahadevan (Ed.), *Robot Learning*. pp. 141-170, Kluver Academic, Boston.

G. Lakoff (1987) *Women, Fire, and Dangerous Things: What Categories Reveal About the Mind*. The University of Chicago Press, Chicago.

G. Lakoff (1988) Cognitive Semantics. in: U. Eco, M. Santambrogio, and P. Violi (Eds.), *Meaning and Mental Representations*. pp. 119-154, Indiana University Press, Bloomington.

R. Laurini and D. Thompson (1992) *Fundamentals of Spatial Information Systems*. Academic Press, San Diego.

D. Leiser and A. Zilbershatz (1989) The Traveller—A Computational Model of Spatial Network Learning. *Environment and Behavior* 21(4): 435-463.

V. Lifschitz (1995) The Logic of Common Sense. *ACM Computing Surveys* 27(3): 343-345.

B. Liskov and J. Guttag (1986) *Abstraction and Specification in Program Development*. The MIT Press, Cambridge, Massachusetts.

B. Liskov and S. Zilles (1979) An Introduction to Formal Specifications of Data Abstractions. in: R. Yeh (Ed.), *Current Trends in Programming Methodology*. Vol. 1 - Software Specification and Design, pp. 1-32, Prentice-Hall, Englewood Cliffs, NJ.

K. Lynch (1960) *The Image of the City*. MIT Press, Cambridge, Massachusetts.

S. Mac Lane and G. Birkhoff (1999) *Algebra*. AMS Chelsea Publishing, Providence, Rhode Island.

H. Mallot, S. Gillner, S. Steck, and M. Franz (1999) Recognition-Triggered Response and the View-Graph Approach to Spatial Cognition. in: C. Freksa and D. Mark (Eds.), *Spatial Information Theory - Cognitive and Computational Foundations of Geographic Information Science, International Conference COSIT '99*, Stade, Germany, pp. 367-380.

D. Mark (1997) Cognitive Perspectives on Spatial and Spatio-temporal Reasoning. in: M. Craglia and H. Couclelis (Eds.), *Geographic Information Research, Bridging the Atlantic*. pp. 308-319, Taylor & Francis, London, UK.

D. Mark, C. Freksa, S. Hirtle, R. Lloyd, and B. Tversky (1999a) Cognitive models of geographical space. *International Journal of Geographical Information Science* 13(8): 747-774.

D. Mark, B. Smith, and B. Tversky (1999b) Ontology and Geographic Objects: An Empirical Study of Cognitive Categorization. in: C. Freksa and D. Mark (Eds.), *Spatial Information Theory - Cognitive and Computational Foundations of Geographic Information Science, International Conference COSIT '99*, Stade, Germany, pp. 283-298.

L. Mark (1987) Eye height-scaled information about affordances: A study of sitting and stair climbing. *Journal of Experimental Psychology: Human Perception and Performance* 13: 361-370.

G. McCalla, L. Reid, and P. Schneider (1982) Plan Creation, Plan Execution, and Knowledge Acquisition in a Dynamic Microworld. *International Journal of Man-Machine Studies* 16: 89-112.

J. McCarthy (1959) Programs with Common Sense. in: *Teddington Conference on the Mechanisation of Thought Processes*, London, pp. 75-91.

D. McDermott and E. Davis (1984) Planning Routes through Uncertain Territory. *Artificial Intelligence* 22: 107-156.

D. Medak (1999) *Lifestyles - A Paradigm for the Description of Spatiotemporal Databases*. Ph.D. thesis, TU Vienna, Vienna.

G. Michaelson (1989) *An Introduction to Functional Programming through Lambda Calculus*. Addison-Wesley, Wokingham, England.

G. Miller (1990) Wordnet: An Online Lexical Database. *Int. Journal of Lexicography* 3(4): 235-312.

S. Moeser (1988) Cognitive mapping in a complex building. *Environment and Behavior* 20: 21-49.

P. Mollerup (2000/01) The way in to the way out: Signage design at Copenhagen Airports. *Information Design Journal* 10(1): 73-78.

D. Montello (1998) A New Framework for Understanding the Acquisition of Spatial Knowledge in Large-Scale Environments. in: M. Egenhofer and R. Golledge (Eds.), *Spatial and Temporal Reasoning in Geographic Information Systems*. *Spatial Information Systems* pp. 143-154, Oxford University Press, New York.

D. Montello (2000) Gibson's theory. Personal communication.

U. Neisser (1976) *Cognition and Reality - Principles and Implications of Cognitive Psychology.* Freeman, New York.

A. Newell and H. Simon (1988) A Theory of Human Problem Solving. in: A. Collins and E. Smith (Eds.), *Readings in Cognitive Science*. pp. 33-51, Morgan Kaufmann Publishers, Inc., San Mateo.

D. Norman (1988) *The Design of Everyday Things*. Doubleday, New York.

H. Nwana and D. Ndumu (1996) An Introduction to Agent Technology. *BT Technology Journal* 14(4):

H. Nwana and D. Ndumu (1999) A Perspective on Software Agents Research. *The Knowledge Engineering Review* 14.

M. O'Neill (1991) A Biologically Based Model of Spatial Cognition and Wayfinding. *Journal of Environmental Psychology* (11): 299-320.

M. O'Neill (1991a) Effects of signage and floor plan configuration on wayfinding accuracy. *Environment and Behavior* 23: 553-574.

M. O'Neill (1991b) Evaluation of a conceptual model of architectural legibility. *Environment and Behavior* 23: 259-284.

D. Osherson and H. Lasnik, Eds. (1990) *Language: An Invitation to Cognitive Science*. MIT Press, Cambridge, Mass.

J. Piaget and B. Inhelder (1967) *The Child's Conception of Space.* Norton, New York.

R. Picard (1997) *Affective Computing*. MIT Press, Cambridge, Mass.

M. Piff (1991) *Discrete Mathematics - An introduction for software engineers*. Cambridge University Press, Cambridge.

S. Pinker (1997) *How the Mind Works.* W. W. Norton & Company, New York.

J. Rasmussen and A. Pejtersen (1995) Virtual Ecology of Work. in: J. Flack, P. Hancock, J. Caird, and K. Vicente (Eds.), *Global Perspectives on the Ecology of Human-Machine Systems*. 1, pp. 121-156, Lawrence Erlbaum Associates, Hillsdale, New Jersey.

M. Raubal (1997) *Structuring Wayfinding Tasks with Image Schemata*. Master thesis, University of Maine, U.S.A., Orono, ME.

M. Raubal and M. Egenhofer (1998) Comparing the complexity of wayfinding tasks in built environments. *Environment & Planning B* 25(6): 895-913.

M. Raubal, M. Egenhofer, D. Pfoser, and N. Tryfona (1997) Structuring Space with Image Schemata: Wayfinding in Airports as a Case Study. in: S. Hirtle and A. Frank (Eds.), *Spatial Information Theory—A Theoretical Basis for GIS, International Conference COSIT '97, Laurel Highlands, PA. Lecture Notes in Computer Science* 1329, pp. 85-102, Springer, Berlin.

M. Raubal and A. Frank (2000) Multimodal Communication for Wayfinding: Airports as a Case Study. Abstract. in: *4th Swedish Symposium on Multimodal Communication (SSoMC)*, Stockholm, Sweden.

M. Raubal and M. Worboys (1999) A Formal Model of the Process of Wayfinding in Built Environments. in: C. Freksa and D. Mark (Eds.), *Spatial Information Theory - Cognitive and Computational Foundations of Geographic Information Science, International Conference COSIT '99*, Stade, Germany, pp. 381-399.

S. Russell and P. Norvig (1995) *Artificial Intelligence - A Modern Approach*. Prentice-Hall International, London.

A. Seidel (1982) Way-Finding in Public Spaces: The Dallas/Fort Worth, USA Airport. in: *20th International Congress of Applied Psychology*, Edinburgh, Scotland.

C. Shannon and W. Weaver (1949) *The Mathematical Theory of Communication*. The University of Illinois Press, Urbana, Illinois.

R. Shaw and J. Bransford, Ed. (1977) *Perceiving, Acting, and Knowing - Toward an Ecological Psychology*. Lawrence Erlbaum Ass., Hillsdale, New Jersey.

A. Siegel and S. White (1975) The development of spatial representations of large-scale environments. in: H. Reese (Ed.), *Advances in child development and behavior*. 10, pp. 9-55, Academic Press, New York.

B. Smith (1995) On Drawing Lines on a Map. in: A. Frank and W. Kuhn (Eds.), *Spatial Information Theory-A Theoretical Basis for GIS. Lecture Notes in Computer Science* 988, pp. 475-484, Springer, Berlin-Heidelberg-New York.

B. Smith (1999) Les objets sociaux. *Philosophiques* 26(2): 315-347.

B. Smith (2001) Objects and Their Environments: From Aristotle to Ecological Ontology. in: A. Frank, J. Raper, and J.-P. Cheylan (Eds.), *Life and Motion of Socio-economic Units*. *GISDATA Series* 8, pp. 79-97, Taylor & Francis, London.

S. Smyth (1992) A Representational Framework for Route Planning in Space and Time. in: *5th International Symposium on Spatial Data Handling*, Charleston, South Carolina, USA, pp. 692-701.

J. Sowa (1999) Mathematical Background. http://www.bestweb.net/~sowa/misc/mathw.htm.

Standard (1996) Translated articles about fire accident at Duesseldorf airport. Der Standard. Vienna.

S. Thompson (1999) *Haskell - The Craft of Functional Programming*. Addison-Wesley, Harlow, England.

H. Timmermans (1991) *Retail environments and spatial shopping behavior*. Department of Architecture, Building, and Planning, University of Technology, Eindhoven, The Netherlands, Technical Report.

S. Timpf, G. Volta, D. Pollock, and M. Egenhofer (1992) A Conceptual Model of Wayfinding Using Multiple Levels of Abstraction. in: A. Frank, I. Campari, and U. Formentini (Eds.), *GIS - From Space to Territory: Theories and Methods of Spatio-Temporal Reasoning*, Pisa, Italy, pp. 348-367.

E. Tolman (1948) Cognitive maps in rats and men. *Psychological Review* 55: 189-208.

R. Trappl, P. Petta, and S. Payr, Eds. (forthcoming) *Emotions in Humans and Artifacts*. MIT Press, Cambridge, MA, U.S.A.

A. Turing (1950) Computing machinery and intelligence. *Mind* 59: 433-460.

B. Tversky (1990) Where partonomies and taxonomies meet. in: S. Tsohatzidis (Ed.), *Meanings and prototypes: Studies on Linguistic Categorization*. pp. 334-344, Routledge, London.

B. Tversky (1993) Cognitive Maps, Cognitive Collages, and Spatial Mental Model. in: A. Frank and I. Campari (Eds.), *Spatial Information Theory: Theoretical Basis for GIS*. *Lecture Notes in Computer Science* 716, pp. 14-24, Springer Verlag, Heidelberg-Berlin.

E. Vanetti and G. Allen (1988) Communicating Environmental Knowledge: The Spatial Impact of Verbal and Spatial Abilities on the Production and Comprehension of Route Directions. *Environment and Behavior* 20(6): 667-682.

W. Warren (1995) Constructing an Econiche. in: J. Flack, P. Hancock, J. Caird, and K. Vicente (Eds.), *Global Perspectives on the Ecology of Human-Machine Systems*. 1, pp. 121-156, Lawrence Erlbaum Associates, Hillsdale, New Jersey.

J. Weisman (1981) Evaluating architectural legibility: Way-finding in the built environment. *Environment and Behavior* 13: 189-204.

G. Weiss, Ed. (1999) *Multiagent Systems - A Modern Approach to Distributed Artificial Intelligence*. MIT Press, Cambridge, MA.

S. Winter and S. Nittel (forthcoming) Abstraction and Standardisation in the Spatial Domain. *International Journal of Geographical Information Science.*

M. Wooldridge (1999) Intelligent Agents. in: G. Weiss (Ed.), *Multiagent Systems - A Modern Approach to Distributed Artificial Intelligence*. pp. 27-77, MIT Press, Cambridge, MA.

M. Worboys (1999) *Observations and Knowledge*. Department of Computer Science, Keele University, Staffs., England, Technical Report TR99. 01.

B. Zaff (1995) Designing with Affordances in Mind. in: J. Flack, P. Hancock, J. Caird, and K. Vicente (Eds.), *Global Perspectives on the Ecology of Human-Machine Systems*. 1, pp. 121-156, Lawrence Erlbaum Associates, Hillsdale, New Jersey.

APPENDIX

This chapter presents the complete Haskell code with its different modules. We also show the specifications for the test data and the results for the test cases.

Agent

```
module Agent where

import ZeroOne
import Environment
import Subfunctions

--*** AGENT ***

data Agent = Agent AgentId ObsSchema AgentState Strategy
          deriving (Show)

type AgentId = Int
data ObsSchema = ObsSchema Position Time Goal
          deriving (Show)
type Time = Int
data AgentState = AgentState [SpatialSit] PrevPosition IncomingDir Decision
          deriving (Show)
type PrevPosition = Position
type IncomingDir = Int
type Decision = GoToAffordance
data Strategy = Strategy Preferences
          deriving (Show)
type Preferences = [(Direction,Preference)]
type Preference = Int

class ObsSchemas obsSchema where
  getPosition :: obsSchema -> Position
  getTime :: obsSchema -> Time
  getGoal :: obsSchema -> Goal

instance ObsSchemas ObsSchema where
  getPosition (ObsSchema p t g) = p
  getTime (ObsSchema p t g) = t
  getGoal (ObsSchema p t g) = g

class AgentStates agentState where
  getSpatialSits :: agentState -> [SpatialSit]
  getPrevPosition :: agentState -> PrevPosition
  getIncomingDir :: agentState -> IncomingDir
  getDecision :: agentState -> Decision

instance AgentStates AgentState where
  getSpatialSits (AgentState ss pp i d) = ss
  getPrevPosition (AgentState ss pp i d) = pp
  getIncomingDir (AgentState ss pp i d) = i
  getDecision (AgentState ss pp i d) = d

--*** AUXILIARY FUNCTION FOR WAYFINDING STRATEGY 2 ***

class IncomingDirs incomingDir where
  nodeDirToAgentDir :: Direction -> incomingDir -> Direction

instance IncomingDirs IncomingDir where
  nodeDirToAgentDir dir incDir = mod (dir + (4 - incDir)) 8

class Strategies strategy where
  getPreferences :: strategy -> Preferences

instance Strategies Strategy where
  getPreferences (Strategy prefs) = prefs

--*** AGENT-FUNCTIONS FOR SIMULATION ***

class Agents agent where
  getAgentId :: agent -> AgentId
```

```
  getObsSchema :: agent -> ObsSchema
  getAgentState :: agent -> AgentState
  getStrategy :: agent -> Strategy

  nodeDirsToAgentDirs :: agent -> [SpatialSit] -> [SpatialSit]
  agentDirsToPrefs :: agent -> [SpatialSit] -> [SpatialSit]
  nodeToPref :: agent -> [SpatialSit] -> [SpatialSit]
--functions for agent's wayfinding strategy 2;

  isAtGoal :: agent -> Bool
  isNotAtGoal :: agent -> Bool
--functions for agent's wayfinding strategy 1;

  see1 :: Node -> agent -> agent
  see :: Environment -> agent -> agent
  decide :: agent -> agent
  act :: agent -> agent
  takeStep :: Environment -> agent -> agent

  wayfinding :: Environment -> agent -> [agent]
  positionsOfAgent :: Environment -> agent -> [Position]
  findCycle :: Environment -> agent -> [Position]
  simulation :: Environment -> agent -> IO()

instance Agents Agent where
  getAgentId (Agent aid obs state strat) = aid
  getObsSchema (Agent aid obs state strat) = obs
  getAgentState (Agent aid obs state strat) = state
  getStrategy (Agent aid obs state strat) = strat

--*** WAYFINDING STRATEGY 2 ***

  nodeDirsToAgentDirs agent@(Agent aid obs state strat) spatialSits
    = map nodeDirToAgentDir' spatialSits
       where nodeDirToAgentDir' (SpatialSit (Info dir gateSign) aff)
                 = SpatialSit (Info (nodeDirToAgentDir dir (getIncomingDir state)) gateSign) aff

  agentDirsToPrefs agent@(Agent aid obs state strat) spatialSits
    = map lookupPref spatialSits
       where lookupPref (SpatialSit (Info dir gateSign) aff)
                 = SpatialSit (Info (unMaybe (lookup dir (getPreferences strat))) gateSign) aff

  nodeToPref agent = agentDirsToPrefs agent . nodeDirsToAgentDirs agent

--*** WAYFINDING STRATEGY 1 ***

  isAtGoal agent
    = ((getSpatialSits (getAgentState agent))/=[]) &&
    isTypeAtGate(getInfoSign(getInfo(head(getSpatialSits(getAgentState agent))))) &&
    (matchGateSign (getGoal(getObsSchema agent))
    (getInfoSign(getInfo(head(getSpatialSits(getAgentState agent))))))

  isNotAtGoal agent = not (isAtGoal agent)

--*** SEE - DECIDE - ACT ***

  see1 node agent@(Agent aid (ObsSchema p t g) (AgentState ss pp i d) strat)
    = (Agent aid (ObsSchema p ti g) (AgentState ssNext pp iNext d) strat)
       where
       ti = t+1
       ssNext = if (getNState node) == []
           then error ("NO SPATIAL SITUATION AT NODE " ++ showAgentPos (agent)
            ++ "!\n" ++ showAgent (agent))
           else getNState node
       iNext = if pp == unit0 then i else unMaybe (lookup pp (getNMatchDir node))

  see env agent
    = see1 (getNodeAtPos (getEnvNodes env) (getPosition (getObsSchema agent))) agent

  decide agent@(Agent aid (ObsSchema p t g) (AgentState ss pp i d) strat)
    = if isAtGoal agent
       then error ("REACHED GOAL " ++ showAgent (agent))
       else (Agent aid (ObsSchema p ti g) (AgentState ss pp i dNext) strat)
         where
         ti = t+1
         dNext = if ss == []
                 then unit0
                 else if (filter ((matchGateSign(g)) . getInfoSign . getInfo) ss) == []
                       then error ("NO MATCHING SIGN INFORMATION AT NODE "
                                 ++ showAgentPos (agent) ++ "!\n" ++ showAgent (agent))
                       else (getAff (head (sortSpatialSits (nodeToPref agent
                                 (filter ((matchGateSign(g)) . getInfoSign . getInfo) ss)))))
```

```
  act (Agent aid (ObsSchema p t g) (AgentState ss pp i d) strat)
    = (Agent aid (ObsSchema pNext ti g) (AgentState ssNext ppNext i dNext) strat)
       where
       ti = t+1
       (pNext,ssNext) = if d==unit0 then (p,ss) else (getEnd(d),[])
       ppNext = p
       dNext = unit0

  takeStep env agent = (act . decide . (see env)) agent

--*** WAYFINDING SIMULATION ***

  wayfinding env agent = take (complexity env) (iterate (takeStep env) agent)

  positionsOfAgent env agent
    = skeletOfList (map getPosition (map getObsSchema (wayfinding env agent)))

  findCycle env agent = findCycle1 (findRepeatingCycle (positionsOfAgent env agent))

  simulation env agent = output where
    output = putStrLn ((showTitle env agent) ++ concat (map showAgent (wayfinding env agent))
                ++ (showCycle (findCycle env agent)) ++ showCheckNodes
                ++ concat (map showNode (map (getNodeAtPos (getEnvNodes env))
                (init (findCycle env agent)))))

--*** INSTANCES FOR ZeroOne CLASS ***

instance ZeroOne Position where
  unit0 = 0

instance ZeroOne GoToAffordance where
  unit0 = (0,0)
  unit100 = (100,100)

--*** TEXT OUTPUT ***

showTitle :: Environment -> Agent -> String
showTitle env agent = "\n*********************************" ++
                      "\nAGENT-BASED WAYFINDING SIMULATION" ++
                      "\n© 2001 by Martin Raubal" ++
                      "\n*********************************" ++ "\n"
  ++ "\nENVIRONMENT: " ++ show (getEnvName env) ++ "\nAGENT " ++ showAgentId agent
  ++ " needs to find " ++ showAgentGoal agent ++ ".\n" ++ "\n//START SIMULATION//" ++ "\n"

showAgent :: Agent -> String
showAgent (Agent aid (ObsSchema p t g) (AgentState ss pp i d) strat)
   = "\nAGENT " ++ show aid ++ "\n OBSERVATION SCHEMA: " ++ "Pos. = " ++ show p ++ ", Time = "
   ++ show t ++ ", Goal = " ++ showGate g ++ "\n AGENT STATE: " ++ "Spatial Sits. = "
   ++ showSpatialSits ss ++ "; Prev. Pos. = " ++ show pp ++ ", Inc. Dir. = " ++ show i
   ++ ", Decision = " ++ showAff d ++ "\n STRATEGY: " ++ showStrat strat ++ "\n"

showSpatialSits :: [SpatialSit] -> String
showSpatialSits [] = "none"
showSpatialSits ss = concat (map showSpatialSit ss)

showSpatialSit :: SpatialSit -> String
showSpatialSit (SpatialSit info aff)
   = "(Info: " ++ showInfo info ++ ", GoTo: " ++ showAff aff ++ ") "

showAff :: GoToAffordance -> String
showAff aff = if aff==unit0 then "none" else (show (getStart aff) ++ "->" ++ show (getEnd aff))

showInfo :: Info -> String
showInfo (Info dir sign) = show dir ++ " - " ++ showGateSign sign

showGateSign :: GateSign -> String
showGateSign gs = if gs==unit0 then "none" else showGateSign1 gs

showGateSign1 :: GateSign -> String
showGateSign1 (GateSign gss) = showGateSignSingle gss
showGateSign1 (GateSign1 gsl) = showGateSignList gsl
showGateSign1 (GateSign2 gsr) = showGateSignRange gsr

showGateSignSingle :: GateSignSingle -> String
showGateSignSingle (GateSignSingle lo) = showLetterOnly lo
showGateSignSingle (GateSignSingle1 g) = showGate g
showGateSignSingle (GateSignSingle2 ag) = showAtGate ag

showGateSignList :: GateSignList -> String
showGateSignList (GateSignList los) = show (map showLetterOnly los)
showGateSignList (GateSignList1 gs) = show gs
```

```
showGateSignRange :: GateSignRange -> String
showGateSignRange (GateSignRange lo1 lo2) = show lo1 ++ "-" ++ show lo2
showGateSignRange (GateSignRange1 g1 g2) = show g1 ++ "-" ++ show g2

showLetterOnly :: LetterOnly -> String
showLetterOnly lo = show (getLetterOnlyLetter lo)

showGate :: Gate -> String
showGate g = show (getGateLetter g) ++ show (getGateNumber g)

showAtGate :: AtGate -> String
showAtGate (AtGate l n) = "at gate " ++ show l ++ show n

showStrat :: Strategy -> String
showStrat (Strategy prefs) = show prefs

outputAgent :: Agent -> IO()
outputAgent agent = putStrLn (showAgent agent)

showAgentId :: Agent -> String
showAgentId (Agent aid (ObsSchema p t g) (AgentState ss pp i d) strat) = show aid

showAgentPos :: Agent -> String
showAgentPos (Agent aid (ObsSchema p t g) (AgentState ss pp i d) strat) = show p

showAgentGoal :: Agent -> String
showAgentGoal (Agent aid (ObsSchema p t g) (AgentState ss pp i d) strat) = show g

showNode :: Node -> String
showNode (Node p s md) = "\nNODE " ++ show p ++ "\n NODE STATE: " ++ show s
                         ++ "\n MATCH DIRECTIONS: " ++ show md ++ "\n"

showCycle :: [Position] -> String
showCycle positions = "\n================================" ++
                      "\nAgent has been caught in a loop: " ++ show positions ++
                      "\n================================" ++ "\n"

showCheckNodes :: String
showCheckNodes = "\nSign information for goal pointing the wrong way at one of these nodes!\n"
```

Environment

```
module Environment where

import ZeroOne

--*** ENVIRONMENT ***

data Environment = Environment Name [Node]
          deriving (Show)

type Name = String
data Node = Node Position NodeState MatchDirection
          deriving (Show)

type Position = Int
type NodeState = [SpatialSit]
type MatchDirection = [(Position,Direction)]
type Direction = Int

class Environments environment where
  getEnvName :: environment -> Name
   getEnvNodes :: environment -> [Node]

  complexity :: environment -> Int

instance Environments Environment where
  getEnvName (Environment n ns) = n
  getEnvNodes (Environment n ns) = ns

  complexity env = length (dropDuplicateEdges (nodesToAffs (getEnvNodes env)))

class Nodes node where
  getNPos :: node -> Position
  getNState :: node -> NodeState
  getNMatchDir :: node -> MatchDirection

  getNodeAtPos :: [node] -> Position -> node
```

```
  nodesToAffs :: [node] -> [GoToAffordance]

instance Nodes Node where
  getNPos (Node p s md) = p
  getNState (Node p s md) = s
  getNMatchDir (Node p s md) = md

  getNodeAtPos ns pos = (head . filter ((pos== ).getNPos)) ns

  nodesToAffs ns = map getAff (concat (map getNState ns))

--*** SPATIAL SITUATIONS ***

data SpatialSit = SpatialSit Info GoToAffordance
          deriving (Show,Eq)

data Info = Info Direction GateSign
          deriving (Show,Eq)

class Infos info where
  getInfoDir :: info -> Direction
  getInfoSign :: info -> GateSign

instance Infos Info where
  getInfoDir (Info dir gs) = dir
  getInfoSign (Info dir gs) = gs

type GoToAffordance = (Position,Position)

--*** COMPLEXITY OF ENVIRONMENT ***

class GoToAffordances goToAffordance where
  getStart :: goToAffordance -> Position
  getEnd :: goToAffordance -> Position

  sameEdge :: goToAffordance -> goToAffordance -> Bool
  dropDuplicateEdges :: [goToAffordance] -> [goToAffordance]
--the functions to determine the complexity value;

instance GoToAffordances GoToAffordance where
  getStart aff = fst (aff)
  getEnd aff = snd (aff)

  sameEdge (a,b) (c,d) = (a==c && b==d) || (a==d && b==c)
--checks if 2 "goTo" affordances belong to the same edge;

  dropDuplicateEdges affs = foldr drop [] affs where
    drop a affs = if any (sameEdge a) affs then affs else a:affs

class SpatialSits spatialSit where
  getInfo :: spatialSit -> Info
  getAff  :: spatialSit -> GoToAffordance

  orderSpatialSits :: spatialSit -> spatialSit -> Bool
  sortSpatialSits  :: [spatialSit] -> [spatialSit]
--functions used in agent's additional wayfinding strategy;

instance SpatialSits SpatialSit where
  getInfo (SpatialSit info aff) = info
  getAff (SpatialSit info aff) = aff

  orderSpatialSits s1 s2 = (getInfoDir . getInfo) s1 <= (getInfoDir . getInfo) s2
--sorts 2 SpatialSits according to preference; this function is input to sort function
--sortSpatialSits;

  sortSpatialSits [] = []
  sortSpatialSits (p:ps) = sortSpatialSits smaller ++ [p] ++ sortSpatialSits larger
                      where
                      smaller = [ q | q<-ps , orderSpatialSits q p ]
                      larger  = [ q | q<-ps , orderSpatialSits p q ]
--sorts a list of SpatialSits according to order of preference;

--*** GATE-SIGNS ***

data GateSign = GateSign GateSignSingle | GateSign1 GateSignList | GateSign2 GateSignRange
          deriving (Show, Eq)

data LetterOnly  = LetterOnly Char
          deriving (Show,Eq)

class LetterOnlys letterOnly where
```

```
  getLetterOnlyLetter :: letterOnly -> Char

instance LetterOnlys LetterOnly where
  getLetterOnlyLetter (LetterOnly l) = l

data Gate = Gate Char Int
          deriving (Show,Eq)
data AtGate = AtGate Char Int
          deriving (Show,Eq)

class Gates gate where
  getGateLetter :: gate -> Char
  getGateNumber :: gate -> Int
  getGate :: gate -> (Char,Int)

instance Gates Gate where
  getGateLetter (Gate l n) = l
  getGateNumber (Gate l n) = n
  getGate (Gate l n) = (l,n)

data GateSignSingle = GateSignSingle Let terOnly | GateSignSingle1 Gate | GateSignSingle2 AtGate
          deriving (Show,Eq)
data GateSignList = GateSignList [LetterOnly] | GateSignList1 [Gate]
          deriving (Show,Eq)
data GateSignRange = GateSignRange LetterOnly LetterOnly | GateSignRange1 Gate Gate
          deriving (Show,Eq)

type Goal = Gate

class GateSigns gateSign where
  matchGateSign :: Goal -> gateSign -> Bool
  isTypeAtGate :: gateSign -> Bool

instance GateSigns GateSign where
  matchGateSign (Gate goal_l goal_n) (GateSign (GateSignSingle (LetterOnly sign_l)))
    = (goal_l==sign_l)
  matchGateSign (Gate goal_l goal_n) (GateSign (GateSignSingle1 (Gate sign_l sign_n)))
    = (goal_l==sign_l) && (goal_n==sign_n)
  matchGateSign (Gate goal_l goal_n) (GateSign (GateSignSingle2 (AtGate sign_l sign_n)))
    = (goal_l==sign_l) && (goal_n==sign_n)
  matchGateSign (Gate goal_l goal_n) (GateSign1 (GateSignList onlyLetters))
    = elem goal_l (map getLetterOnlyLetter onlyLetters)
  matchGateSign goal (GateSign1 (GateSignList1 gates))
    = elem goal gates
  matchGateSign (Gate goal_l goal_n) (GateSign2 (GateSignRange onlyLetter1 onlyLetter2))
    = ((goal_l)>=getLetterOnlyLetter(onlyLetter1)) &&
      ((goal_l)<=getLetterOnlyLetter(onlyLetter2))
  matchGateSign (Gate goal_l goal_n) (GateSign2 (GateSignRange1 gate1 gate2))
    = (goal_l==getGateLetter(gate1)) && (goal_l==getGateLetter(gate2)) &&
      (goal_n>=getGateNumber(gate1)) && (goal_n<=getGateNumber(gate2))
  matchGateSign goal sign
    = error ("no match between goal and gatesign " ++ show goal ++ show sign)

  isTypeAtGate (GateSign (GateSignSingle2 (AtGate l n))) = True
  isTypeAtGate _ = False

--*** INSTANCES FOR ZeroOne CLASS ***

instance ZeroOne GateSign where
  unit0 = GateSign unit0

instance ZeroOne GateSignSingle where
  unit0 = GateSignSingle unit0

instance ZeroOne LetterOnly where
  unit0 = LetterOnly unit0
```

Subfunctions

```
module Subfunctions where

--*** AUXILIARY FUNCTIONS FOR findCycle ***

--following are the subfunctions for the findCycle function defined in the class Agents;
skeletOfList :: Eq a => [a] -> [a]
skeletOfList [] = []
skeletOfList [a] = [a]
skeletOfList (a : xa) = if (a == head (xa)) then skeletOfList (xa) else (a : skeletOfList (xa))
--the result of this function is the skeleton of a list;
```

```
matches :: Int -> [Int] -> [Int]
matches int listOfIntegers = filter (int==) listOfIntegers
--this function picks out all occurences of an integer in a list;

findRepeatingCycle :: [Int] -> [Int]
findRepeatingCycle (a : as) = if (length (matches a as) > 1)
                              then (a : as) else findRepeatingCycle as
--gives a list with the repeating cycle;

findCycle1 :: [Int] -> [Int]
findCycle1 [] = []
findCycle1 (a : xa) = (a : takeWhile (a /=) (xa)) ++ (a : [])
--gives a list with one occurence of the cycle;

--*** OTHER AUXILIARY FUNCTIONS ***

unMaybe :: Maybe a -> a
unMaybe (Just a) = a
unMaybe Nothing = error ("unMaybe of Nothing")
```

ZeroOne

```
module ZeroOne where

class ZeroOne z where
  unit0, unit1, unit100 :: z

  isZero, notZero, isOne :: Eq z => z -> Bool
  lessZero, greaterZero :: Ord z => z -> Bool
  lessZero a = a <= unit0
  greaterZero a = a >= unit0
  isZero a = unit0 == a
  isOne a = unit1 == a
  notZero = not.isZero

instance ZeroOne Bool where
  unit0 = False
  unit1 = True

--instance ZeroOne Int where
--  unit0 = 0
--  unit1 = 1
--  unit100 = 100

instance ZeroOne Float where
  unit0 = 0.0
  unit1 = 1.0

instance ZeroOne Char where
  unit0 = ' '
  unit1 = '\n'

instance ZeroOne String where
  unit0 = ""

--instance ZeroOne [a] where
--  unit0 = []

instance (ZeroOne a, ZeroOne b) => ZeroOne (a,b) where
  unit0 = (unit0, unit0)
  unit1 = (unit1, unit1)
  unit100 = (unit100, unit100)

instance (ZeroOne a, ZeroOne b, ZeroOne c) => ZeroOne (a,b,c) where
  unit0 = (unit0, unit0, unit0)
  unit1 = (unit1, unit1, unit1)
```

Test data for the environment

```
module VIETestDataComplete where

import Environment
import Agent
import ZeroOne
```

```
node1,node2,node3,node4,node5,node6,node7,node8,node9,node10,node11,node12,node13,node14,node15,
node16,node17,node18,node19,node20,node21,node22,node23,node24,node25,node26,node27,node28,
node29,node30,node31,node32,node33,node34,node35,node36,node37,node38,node39,node40,node41,
node42,node43,node44,node45 :: Node

node1 = Node 1 [SpatialSit (Info 2 (GateSign1 (GateSignList [(LetterOnly 'A'),(LetterOnly
'B'),(LetterOnly 'C')]))) (1,2)] [(0,7)]

node2 = Node 2 [SpatialSit (Info 0 (GateSign1 (GateSignList [(LetterOnly 'A'),(LetterOnly
'B'),(LetterOnly 'C')]))) (2,3)] [(1,6)]

node3 = Node 3 [SpatialSit (Info 0 (GateSign1 (GateSignList [(LetterOnly 'A'),(LetterOnly
'C')]))) (3,5),SpatialSit (Info 1 (GateSign (GateSignSingle (LetterOnly 'A')))) (3,4),SpatialSit
(Info 6 (GateSign1 (GateSignList [(LetterOnly 'B'),(LetterOnly 'C')]))) (3,6)]
[(2,4),(4,1),(5,0),(6,6)]

node4 = Node 4 [SpatialSit (Info 2 (GateSign (GateSignSingle (LetterOnly 'A'))))
(4,32),SpatialSit (Info 3 unit0) (4,31),SpatialSit (Info 5 unit0) (4,3),SpatialSit (Info 6
(GateSign1 (GateSignList [(LetterOnly 'B'),(LetterOnly 'C')]))) (4,5)]
[(3,5),(5,6),(31,3),(32,2)]

--manipulated node 4 so that agent gets caught in a loop;
--changing signs for directions 2 and 5; therefore sign for gate area A points back to node 3!
node4 = Node 4 [SpatialSit (Info 2 unit0) (4,32),SpatialSit (Info 3 unit0) (4,31),SpatialSit
(Info 5 (GateSign (GateSignSingle (LetterOnly 'A')))) (4,3), SpatialSit (Info 6 (GateSign1
(GateSignList [(LetterOnly 'B'),(LetterOnly 'C')]))) (4,5)] [(3,5),(5,6),(31,3),(32,2)]

node5 = Node 5 [SpatialSit (Info 2 (GateSign (GateSignSingle (LetterOnly 'A')))) (5,4),SpatialSit
(Info 4 unit0) (5,3),SpatialSit (Info 6 (GateSign (GateSignSingle (LetterOnly 'B'))))
(5,7),SpatialSit (Info 6 (GateSign (GateSignSingle (LetterOnly 'C')))) (5,7)] [(3,4),(4,2),(7,6)]

node6 = Node 6 [SpatialSit (Info 0 (GateSign (GateSignSingle (LetterOnly 'C')))) (6,7),SpatialSit
(Info 3 unit0) (6,3),SpatialSit (Info 6 (GateSign (GateSignSingle (LetterOnly 'B')))) (6,26)]
[(3,3),(7,0),(26,6)]

node7 = Node 7 [SpatialSit (Info 2 (GateSign (GateSignSingle (LetterOnly 'A')))) (7,5),SpatialSit
(Info 4 (GateSign (GateSignSingle (LetterOnly 'B')))) (7,6),SpatialSit (Info 6 (GateSign
(GateSignSingle (LetterOnly 'C')))) (7,8)] [(5,2),(6,4),(8,6)]

node8 = Node 8 [SpatialSit (Info 3 (GateSign1 (GateSignList [(LetterOnly 'A'),(LetterOnly
'B')]))) (8,7),SpatialSit (Info 7 (GateSign2 (GateSignRange1 (Gate 'C' 51) (Gate 'C' 62))))
(8,9)] [(7,3),(9,7)]

node9 = Node 9 [SpatialSit (Info 0 (GateSign2 (GateSignRange1 (Gate 'C' 52) (Gate 'C' 62))))
(9,11),SpatialSit (Info 3 (GateSign1 (GateSignList [(LetterOnly 'A'),(LetterOnly 'B')])))
(9,8),SpatialSit (Info 6 (GateSign (GateSignSingle1 (Gate 'C' 51)))) (9,10)] [(8,3),(11,0)]

node10 = Node 10 [SpatialSit (Info unit0 (GateSign (GateSignSingle2 (AtGate 'C' 51)))) unit0]
[(9,2)]

node11 = Node 11 [SpatialSit (Info 0 unit0) (11,13),SpatialSit (Info 2 (GateSign (GateSignSingle1
(Gate 'C' 62)))) (11,12),SpatialSit (Info 4 (GateSign1 (GateSignList [(LetterOnly
'A'),(LetterOnly 'B')]))) (11,9)] [(9,4),(13,0)]

node12 = Node 12 [SpatialSit (Info unit0 (GateSign (GateSignSingle2 (AtGate 'C' 62)))) unit0]
[(11,6)]

node13 = Node 13 [SpatialSit (Info 1 (GateSign2 (GateSignRange1 (Gate 'C' 54) (Gate 'C' 61))))
(13,17),SpatialSit (Info 4 (GateSign1 (GateSignList [(LetterOnly 'A'),(LetterOnly 'B')])))
(13,11),SpatialSit (Info 6 (GateSign1 (GateSignList1 [(Gate 'C' 52),(Gate 'C' 53)]))) (13,14)]
[(11,4),(17,1)]

node14 = Node 14 [SpatialSit (Info 5 (GateSign (GateSignSingle1 (Gate 'C' 52))))
(14,15),SpatialSit (Info 7 (GateSign (GateSignSingle1 (Gate 'C' 53)))) (14,16)] [(13,2)]

node15 = Node 15 [SpatialSit (Info unit0 (GateSign (GateSignSingle2 (AtGate 'C' 52)))) unit0]
[(14,2)]

node16 = Node 16 [SpatialSit (Info unit0 (GateSign (GateSignSingle2 (AtGate 'C' 53)))) unit0]
[(14,3)]

node17 = Node 17 [SpatialSit (Info 0 (GateSign (GateSignSingle1 (Gate 'C' 57))))
(17,21),SpatialSit (Info 1 (GateSign (GateSignSingle1 (Gate 'C' 58)))) (17,22),SpatialSit (Info 1
(GateSign (GateSignSingle1 (Gate 'C' 59)))) (17,23),SpatialSit (Info 2 (GateSign (GateSignSingle1
(Gate 'C' 60)))) (17,24),SpatialSit (Info 2 (GateSign (GateSignSingle1 (Gate 'C' 61))))
(17,25),SpatialSit (Info 5 (GateSign1 (GateSignList [(LetterOnly 'A'),(LetterOnly 'B')])))
(17,13),SpatialSit (Info 6 (GateSign (GateSignSingle1 (Gate 'C' 54)))) (17,18),SpatialSit (Info 6
(GateSign (GateSignSingle1 (Gate 'C' 55)))) (17,19),SpatialSit (Info 7 (GateSign (GateSignSingle1
(Gate 'C' 56)))) (17,20)] [(13,5)]

node18 = Node 18 [SpatialSit (Info unit0 (GateSign (GateSignSingle2 (AtGate 'C' 54)))) unit0]
[(17,2)]
```

```
node19 = Node 19 [SpatialSit (Info unit0 (GateSign (GateSignSingle2 (AtGate 'C' 55)))) unit0]
[(17,3)]

node20 = Node 20 [SpatialSit (Info unit0 (GateSign (GateSignSingle2 (AtGate 'C' 56)))) unit0]
[(17,4)]

node21 = Node 21 [SpatialSit (Info unit0 (GateSign (GateSignSingle2 (AtGate 'C' 57)))) unit0]
[(17,5)]

node22 = Node 22 [SpatialSit (Info unit0 (GateSign (GateSignSingle2 (AtGate 'C' 58)))) unit0]
[(17,5)]

node23 = Node 23 [SpatialSit (Info unit0 (GateSign (GateSignSingle2 (AtGate 'C' 59)))) unit0]
[(17,6)]

node24 = Node 24 [SpatialSit (Info unit0 (GateSign (GateSignSingle2 (AtGate 'C' 60)))) unit0]
[(17,6)]

node25 = Node 25 [SpatialSit (Info unit0 (GateSign (GateSignSingle2 (AtGate 'C' 61)))) unit0]
[(17,7)]

node26 = Node 26 [SpatialSit (Info 2 (GateSign1 (GateSignList [(LetterOnly 'A'),(LetterOnly
'C')]))) (26,6),SpatialSit (Info 6 (GateSign (GateSignSingle (LetterOnly 'B')))) (26,27)]
[(6,2),(27,6)]

node27 = Node 27 [SpatialSit (Info 2 unit0) (27,26),SpatialSit (Info 6 (GateSign2 (GateSignRange1
(Gate 'B' 25) (Gate 'B' 43)))) (27,29),SpatialSit (Info 7 (GateSign2 (GateSignRange1 (Gate 'B'
22) (Gate 'B' 24)))) (27,28)] [(26,2),(29,6)]

node28 = Node 28 [] [(27,3)]

node29 = Node 29 [SpatialSit (Info 2 unit0) (29,27)] [(27,2)]

node30 = Node 30 [SpatialSit (Info 2 (GateSign1 (GateSignList [(LetterOnly 'A'),(LetterOnly
'B'),(LetterOnly 'C')]))) (30,31)] [(0,7)]

node31 = Node 31 [SpatialSit (Info 0 (GateSign (GateSignSingle (LetterOnly 'A'))))
(31,32),SpatialSit (Info 1 (GateSign (GateSignSingle (LetterOnly 'A')))) (31,32),SpatialSit (Info
6 (GateSign1 (GateSignList [(LetterOnly 'B'),(LetterOnly 'C')]))) (31,4)]
[(4,6),(30,4),(32,0),(32,1)]

node32 = Node 32 [SpatialSit (Info 1 (GateSign2 (GateSignRange1 (Gate 'A' 1) (Gate 'A' 19))))
(32,33),SpatialSit (Info 4 unit0) (32,31),SpatialSit (Info 5 unit0) (32,31),SpatialSit (Info 6
(GateSign1 (GateSignList [(LetterOnly 'B'),(LetterOnly 'C')]))) (32,4)] [(4,6),(31,4),(31,5)]

node33 = Node 33 [SpatialSit (Info 2 (GateSign2 (GateSignRange1 (Gate 'A' 1) (Gate 'A' 19))))
(33,35),SpatialSit (Info 4 (GateSign1 (GateSignList [(LetterOnly 'B'),(LetterOnly 'C')])))
(33,34)] [(32,6),(35,2)]

node34 = Node 34 [] [(33,7)]

node35 = Node 35 [SpatialSit (Info 1 (GateSign2 (GateSignRange1 (Gate 'A' 1) (Gate 'A' 8))))
(35,37),SpatialSit (Info 2 (GateSign2 (GateSignRange1 (Gate 'A' 10) (Gate 'A' 19))))
(35,36),SpatialSit (Info 5 (GateSign1 (GateSignList [(LetterOnly 'B'),(LetterOnly 'C')])))
(35,33)] [(33,5),(37,1)]

node36 = Node 36 [] [(35,7)]

node37 = Node 37 [SpatialSit (Info 0 (GateSign (GateSignSingle1 (Gate 'A' 5))))
(37,42),SpatialSit (Info 1 (GateSign (GateSignSingle1 (Gate 'A' 4)))) (37,41),SpatialSit (Info 1
(GateSign (GateSignSingle1 (Gate 'A' 3)))) (37,40),SpatialSit (Info 2 (GateSign (GateSignSingle1
(Gate 'A' 2)))) (37,39),SpatialSit (Info 2 (GateSign (GateSignSingle1 (Gate 'A' 1))))
(37,38),SpatialSit (Info 4 (GateSign1 (GateSignList [(LetterOnly 'B'),(LetterOnly 'C')])))
(37,35),SpatialSit (Info 4 (GateSign2 (GateSignRange1 (Gate 'A' 10) (Gate 'A' 19))))
(37,35),SpatialSit (Info 6 (GateSign (GateSignSingle1 (Gate 'A' 8)))) (37,45),SpatialSit (Info 7
(GateSign (GateSignSingle1 (Gate 'A' 7)))) (37,44),SpatialSit (Info 7 (GateSign (GateSignSingle1
(Gate 'A' 6)))) (37,43)] [(35,4)]

node38 = Node 38 [SpatialSit (Info unit0 (GateSign (GateSignSingle2 (AtGate 'A' 1)))) unit0]
[(37,6)]

node39 = Node 39 [SpatialSit (Info unit0 (GateSign (GateSignSingle2 (AtGate 'A' 2)))) unit0]
[(37,6)]

node40 = Node 40 [SpatialSit (Info unit0 (GateSign (GateSignSingle2 (AtGate 'A' 3)))) unit0]
[(37,5)]

node41 = Node 41 [SpatialSit (Info unit0 (GateSign (GateSignSingle2 (AtGate 'A' 4)))) unit0]
[(37,5)]
```

```
node42 = Node 42 [SpatialSit (Info unit0 (GateSign (GateSignSingle2 (AtGate 'A' 5)))) unit0]
[(37,4)]

node43 = Node 43 [SpatialSit (Info unit0 (GateSign (GateSignSingle2 (AtGate 'A' 6)))) unit0]
[(37,3)]

node44 = Node 44 [SpatialSit (Info unit0 (GateSign (GateSignSingle2 (AtGate 'A' 7)))) unit0]
[(37,3)]

node45 = Node 45 [SpatialSit (Info unit0 (GateSign (GateSignSingle2 (AtGate 'A' 8)))) unit0]
[(37,2)]

vie :: Environment
vie = Environment "Vienna Int. Airport"
[node1,node2,node3,node4,node5,node6,node7,node8,node9,node10,node11,node12,node13,node14,node15,
node16,node17,node18,node19,node20,node21,node22,node23,node24,node25,node26,node27,node28,
node29,node30,node31,node32,node33,node34,node35,node36,node37,node38,node39,node40,node41,
node42,node43,node44,node45]
```

Test data for the agent

```
pref :: Preferences
pref = [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

agent1 :: Agent
agent1 = Agent 1 (ObsSchema 1 1 (Gate 'A' 6)) (AgentState [] unit0 7 unit0) (Strategy pref)
--agent reaches goal

agent2 :: Agent
agent2 = Agent 2 (ObsSchema 30 1 (Gate 'C' 56)) (AgentState [] unit0 7 unit0) (Strategy pref)
--no matching sign information at node

agent3 :: Agent
agent3 = Agent 3 (ObsSchema 26 1 (Gate 'A' 6)) (AgentState [] unit0 7 unit0) (Strategy pref)
--no matching sign information at node

agent4 :: Agent
agent4 = Agent 4 (ObsSchema 1 1 (Gate 'A' 6)) (AgentState [] unit0 7 unit0) (Strategy pref)
--agent caught in a loop by changing the A-sign at node 4 (directing back to 3)  - not real!
```

Results for test case 1

```
VIETestDataComplete> simulation vie agent1

*********************************
AGENT-BASED WAYFINDING SIMULATION
© 2001 by Martin Raubal
*********************************

ENVIRONMENT: "Vienna Int. Airport"
AGENT 1 needs to find Gate 'A' 6.

//START SIMULATION//

AGENT 1
 OBSERVATION SCHEMA: Pos. = 1, Time = 1, Goal = 'A'6
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 0, Inc. Dir. = 7, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

AGENT 1
 OBSERVATION SCHEMA: Pos. = 2, Time = 4, Goal = 'A'6
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 1, Inc. Dir. = 7, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

AGENT 1
 OBSERVATION SCHEMA: Pos. = 3, Time = 7, Goal = 'A'6
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 2, Inc. Dir. = 6, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

AGENT 1
 OBSERVATION SCHEMA: Pos. = 5, Time = 10, Goal = 'A'6
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 3, Inc. Dir. = 4, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

AGENT 1
 OBSERVATION SCHEMA: Pos. = 4, Time = 13, Goal = 'A'6
```

```
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 5, Inc. Dir. = 4, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

AGENT 1
 OBSERVATION SCHEMA: Pos. = 32, Time = 16, Goal = 'A'6
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 4, Inc. Dir. = 6, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

AGENT 1
 OBSERVATION SCHEMA: Pos. = 33, Time = 19, Goal = 'A'6
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 32, Inc. Dir. = 6, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

AGENT 1
 OBSERVATION SCHEMA: Pos. = 35, Time = 22, Goal = 'A'6
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 33, Inc. Dir. = 6, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

AGENT 1
 OBSERVATION SCHEMA: Pos. = 37, Time = 25, Goal = 'A'6
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 35, Inc. Dir. = 5, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

AGENT 1
 OBSERVATION SCHEMA: Pos. = 43, Time = 28, Goal = 'A'6
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 37, Inc. Dir. = 4, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

Program execution error: REACHED GOAL
AGENT 1
 OBSERVATION SCHEMA: Pos. = 43, Time = 29, Goal = 'A'6
 AGENT STATE: Spatial Sits. = (Info: 0 - at gate 'A'6, GoTo: none) ; Prev. Pos. = 37,
                              Inc. Dir.= 3, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]
```

Results for test case 2

```
VIETestDataComplete> simulation vie agent2

*********************************
AGENT-BASED WAYFINDING SIMULATION
© 2001 by Martin Raubal
*********************************

ENVIRONMENT: "Vienna Int. Airport"
AGENT 2 needs to find Gate 'C' 56.

//START SIMULATION//

AGENT 2
 OBSERVATION SCHEMA: Pos. = 30, Time = 1, Goal = 'C'56
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 0, Inc. Dir. = 7, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

AGENT 2
 OBSERVATION SCHEMA: Pos. = 31, Time = 4, Goal = 'C'56
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 30, Inc. Dir. = 7, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

AGENT 2
 OBSERVATION SCHEMA: Pos. = 4, Time = 7, Goal = 'C'56
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 31, Inc. Dir. = 4, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

AGENT 2
 OBSERVATION SCHEMA: Pos. = 5, Time = 10, Goal = 'C'56
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 4, Inc. Dir. = 3, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

AGENT 2
 OBSERVATION SCHEMA: Pos. = 7, Time = 13, Goal = 'C'56
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 5, Inc. Dir. = 2, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

AGENT 2
 OBSERVATION SCHEMA: Pos. = 8, Time = 16, Goal = 'C'56
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 7, Inc. Dir. = 2, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]
```

```
AGENT 2
 OBSERVATION SCHEMA: Pos. = 9, Time = 19, Goal = 'C'56
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 8, Inc. Dir. = 3, Decision = no ne
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

AGENT 2
 OBSERVATION SCHEMA: Pos. = 11, Time = 22, Goal = 'C'56
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 9, Inc. Dir. = 3, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

AGENT 2
 OBSERVATION SCHEMA: Pos. =
Program execution error: NO MATCHING SIGN INFORMATION AT NODE 11!

AGENT 2
 OBSERVATION SCHEMA: Pos. = 11, Time = 23, Goal = 'C'56
 AGENT STATE: Spatial Sits. = (Info: 0 - none, GoTo: 11->13) (Info: 2 - 'C'62, GoTo: 11->12)
    (Info: 4 - ["'A'","'B'"], GoTo: 11->9) ; Prev. Pos. = 9, Inc. Dir. = 4, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]


VIETestDataComplete> simulation vie agent3

*********************************
AGENT-BASED WAYFINDING SIMULATION
© 2001 by Martin Raubal
*********************************

ENVIRONMENT: "Vienna Int. Airport"
AGENT 3 needs to find Gate 'A' 6.

//START SIMULATION//

AGENT 3
 OBSERVATION SCHEMA: Pos. = 26, Time = 1, Goal = 'A'6
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 0, Inc. Dir. = 7, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

AGENT 3
 OBSERVATION SCHEMA: Pos. = 6, Time = 4, Goal = 'A'6
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 26, Inc. Dir. = 7, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

AGENT 3
 OBSERVATION SCHEMA: Pos. =
Program execution error: NO MATCHING SIGN INFORMATION AT NODE 6!

AGENT 3
 OBSERVATION SCHEMA: Pos. = 6, Time = 5, Goal = 'A'6
 AGENT STATE: Spatial Sits. = (Info: 0 - 'C', GoTo: 6->7) (Info: 3 - none, GoTo: 6->3)
    (Info: 6 - 'B', GoTo: 6->26) ; Prev. Pos. = 26, Inc. Dir. = 6, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]
```

Results for test case 3

```
VIETestDataComplete> simulation vie agent4

*********************************
AGENT-BASED WAYFINDING SIMULATION
© 2001 by Martin Raubal
*********************************

ENVIRONMENT: "Vienna Int. Airport"
AGENT 4 needs to find Gate 'A' 6.

//START SIMULATION//

AGENT 4
 OBSERVATION SCHEMA: Pos. = 1, Time = 1, Goal = 'A'6
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 0, Inc. Dir. = 7, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

AGENT 4
 OBSERVATION SCHEMA: Pos. = 2, Time = 4, Goal = 'A'6
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 1, Inc. Dir. = 7, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

AGENT 4
 OBSERVATION SCHEMA: Pos. = 3, Time = 7, Goal = 'A'6
```

```
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 2, Inc. Dir. = 6, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

AGENT 4
 OBSERVATION SCHEMA: Pos. = 5, Time = 10, Goal = 'A'6
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 3, Inc. Dir. = 4, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

AGENT 4
 OBSERVATION SCHEMA: Pos. = 4, Time = 13, Goal = 'A'6
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 5, Inc. Dir. = 4, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

AGENT 4
 OBSERVATION SCHEMA: Pos. = 3, Time = 16, Goal = 'A'6
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 4, Inc. Dir. = 6, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

AGENT 4
 OBSERVATION SCHEMA: Pos. = 5, Time = 19, Goal = 'A'6
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 3, Inc. Dir. = 1, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

AGENT 4
 OBSERVATION SCHEMA: Pos. = 4, Time = 22, Goal = 'A'6
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 5, Inc. Dir. = 4, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

AGENT 4
 OBSERVATION SCHEMA: Pos. = 3, Time = 25, Goal = 'A'6
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 4, Inc. Dir. = 6, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

AGENT 4
 OBSERVATION SCHEMA: Pos. = 5, Time = 28, Goal = 'A'6
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 3, Inc. Dir. = 1, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

AGENT 4
 OBSERVATION SCHEMA: Pos. = 4, Time = 31, Goal = 'A'6
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 5, Inc. Dir. = 4, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

AGENT 4
 OBSERVATION SCHEMA: Pos. = 3, Time = 34, Goal = 'A'6
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 4, Inc. Dir. = 6, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

AGENT 4
 OBSERVATION SCHEMA: Pos. = 5, Time = 37, Goal = 'A'6
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 3, Inc. Dir. = 1, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

AGENT 4
 OBSERVATION SCHEMA: Pos. = 4, Time = 40, Goal = 'A'6
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 5, Inc. Dir. = 4, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

AGENT 4
 OBSERVATION SCHEMA: Pos. = 3, Time = 43, Goal = 'A'6
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 4, Inc. Dir. = 6, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),( 4,8),(5,7),(6,5),(7,3)]

AGENT 4
 OBSERVATION SCHEMA: Pos. = 5, Time = 46, Goal = 'A'6
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 3, Inc. Dir. = 1, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

AGENT 4
 OBSERVATION SCHEMA: Pos. = 4, Time = 49, Goal = 'A'6
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 5, Inc. Dir. = 4, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

AGENT 4
 OBSERVATION SCHEMA: Pos. = 3, Time = 52, Goal = 'A'6
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 4, Inc. Dir. = 6, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

AGENT 4
 OBSERVATION SCHEMA: Pos. = 5, Time = 55, Goal = 'A'6
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 3, Inc. Dir . = 1, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]
```

```
AGENT 4
 OBSERVATION SCHEMA: Pos. = 4, Time = 58, Goal = 'A'6
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 5, Inc. Dir. = 4, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

AGENT 4
 OBSERVATION SCHEMA: Pos. = 3, Time = 61, Goal = 'A'6
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 4, Inc. Dir. = 6, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

AGENT 4
 OBSERVATION SCHEMA: Pos. = 5, Time = 64, Goal = 'A'6
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 3, Inc. Dir. = 1, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

AGENT 4
 OBSERVATION SCHEMA: Pos. = 4, Time = 67, Goal = 'A'6
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 5, Inc. Dir. = 4, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

AGENT 4
 OBSERVATION SCHEMA: Pos. = 3, Time = 70, Goal = 'A'6
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 4, Inc. Dir. = 6, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

AGENT 4
 OBSERVATION SCHEMA: Pos. = 5, Time = 73, Goal = 'A'6
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 3, Inc. Dir. = 1, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

AGENT 4
 OBSERVATION SCHEMA: Pos. = 4, Time = 76, Goal = 'A'6
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 5, Inc. Dir. = 4, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

AGENT 4
 OBSERVATION SCHEMA: Pos. = 3, Time = 79, Goal = 'A'6
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 4, Inc. Dir. = 6, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

AGENT 4
 OBSERVATION SCHEMA: Pos. = 5, Time = 82, Goal = 'A'6
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 3, Inc. Dir. = 1, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

AGENT 4
 OBSERVATION SCHEMA: Pos. = 4, Time = 85, Goal = 'A'6
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 5, Inc. Dir. = 4, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

AGENT 4
 OBSERVATION SCHEMA: Pos. = 3, Time = 88, Goal = 'A'6
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 4, Inc. Dir. = 6, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

AGENT 4
 OBSERVATION SCHEMA: Pos. = 5, Time = 91, Goal = 'A'6
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 3, Inc. Dir. = 1, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

AGENT 4
 OBSERVATION SCHEMA: Pos. = 4, Time = 94, Goal = 'A'6
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 5, Inc. Dir. = 4, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

AGENT 4
 OBSERVATION SCHEMA: Pos. = 3, Time = 97, Goal = 'A'6
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 4, Inc. Dir. = 6, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

AGENT 4
 OBSERVATION SCHEMA: Pos. = 5, Time = 100, Goal = 'A'6
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 3, Inc. Dir. = 1, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

AGENT 4
 OBSERVATION SCHEMA: Pos. = 4, Time = 103, Goal = 'A'6
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 5, Inc. Dir. = 4, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

AGENT 4
```

```
 OBSERVATION SCHEMA: Pos. = 3, Time = 106, Goal = 'A'6
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 4, Inc. Dir. = 6, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

AGENT 4
 OBSERVATION SCHEMA: Pos. = 5, Time = 109, Goal = 'A'6
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 3, Inc. Dir. = 1, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

AGENT 4
 OBSERVATION SCHE MA: Pos. = 4, Time = 112, Goal = 'A'6
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 5, Inc. Dir. = 4, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

AGENT 4
 OBSERVATION SCHEMA: Pos. = 3, Time = 115, Goal = 'A'6
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 4, Inc. Dir. = 6, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

AGENT 4
 OBSERVATION SCHEMA: Pos. = 5, Time = 118, Goal = 'A'6
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 3, Inc. Dir. = 1, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

AGENT 4
 OBSERVATION SCHEMA: Pos. = 4, Time = 121, Goal = 'A'6
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 5, Inc. Dir. = 4, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

AGENT 4
 OBSERVATION SCHEMA: Pos. = 3, Time = 124, Goal = 'A'6
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 4, Inc. Dir. = 6, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

AGENT 4
 OBSERVATION SCHEMA: Pos. = 5, Time = 127, Goal = 'A'6
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 3, Inc. Dir. = 1, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

AGENT 4
 OBSERVATION SCHEMA: Pos. = 4, Time = 130, Goal = 'A'6
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 5, Inc. Dir. = 4, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

AGENT 4
 OBSERVATION SCHEMA: Pos. = 3, Time = 133, Goal = 'A'6
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 4, Inc. Dir. = 6, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

AGENT 4
 OBSERVATION SCHEMA: Pos. = 5, Time = 136, Goal = 'A'6
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 3, Inc. Dir. = 1, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

AGENT 4
 OBSERVATION SCHEMA: Pos. = 4, Time = 139, Goal = 'A'6
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 5, Inc. Dir. = 4, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

AGENT 4
 OBSERVATION SCHEMA: Pos. = 3, Time = 142, Goal = 'A'6
 AGENT STATE: Spatial Sits. = none; Prev. Pos. = 4, Inc. Dir. = 6, Decision = none
 STRATEGY: [(0,1),(1,2),(2,4),(3,6),(4,8),(5,7),(6,5),(7,3)]

================================
Agent has been caught in a loop: [3,5,4,3]
================================

Sign information for goal pointing the wrong way at one of these nodes!

NODE 3
 NODE STATE: [SpatialSit (Info 0 (GateSign1 (GateSignList [LetterOnly 'A',LetterOnly 'C'])))
    (3,5),SpatialSit (Info 1 (GateSign (GateSignSingle (LetterOnly 'A')))) (3,4),SpatialSit
    (Info 6 (GateSign1 (GateSignList [LetterOnly 'B',LetterOnly 'C']))) (3,6)]
 MATCH DIRECTIONS: [(2,4),(4,1),(5,0),(6,6)]

NODE 5
 NODE STATE: [SpatialSit (Info 2 (GateSign (GateSignSingle (LetterOnly 'A')))) (5,4),
    SpatialSit (Info 4 (GateSign (GateSignSingle (LetterOnly ' ')))) (5,3),SpatialSit
    (Info 6 (GateSign (GateSignSingle (LetterOnly 'B')))) (5,7),SpatialSit (Info 6
    (GateSign (GateSignSingle (LetterOnly 'C')))) (5,7)]
 MATCH DIRECTIONS: [(3,4),(4,2),(7,6)]
```

```
NODE 4
 NODE STATE: [SpatialSit (Info 2 (GateSign (GateSignSingle (LetterOnly ' '))) (4,32),
    SpatialSit (Info 3 (GateSign (GateSignSingle (LetterOnly ' '))) (4,31),
    SpatialSit (Info 5 (GateSign (GateSignSingle (LetterOnly 'A'))) (4,3),
    SpatialSit (Info 6 (GateSign1 (GateSignList [LetterOnly'B',LetterOnly 'C']))) (4,5)]
 MATCH DIRECTIONS: [(3,5),(5,6),(31,3),(32,2)]
```

BIOGRAPHY OF THE AUTHOR

Martin Raubal was born in Vienna, Austria on September 11, 1968. He received his high school diploma with honors from the Stiftsgymnasium Melk, Austria in 1986. After serving in the Austrian Army he entered the Vienna University of Technology in 1987. Between 1987 and 1990 Martin worked for the Austrian Surveying Agency for several months. In 1991/92 he studied English at the Cambridge English Language Centers in Sydney, Australia, and received a diploma. After returning to Austria he worked part-time as a surveyor from 1993 - 1995 and received the "1st Diploma" in Surveying Engineering in March 1995. In 1995 and 1996 Martin worked as a Teaching Assistant at the Department of Geoinformation, Vienna University of Technology.

In the fall of 1996 he entered the master's degree program at the University of Maine (U.S.A.) in the Department of Spatial Information Science and Engineering where he also worked as a Research Assistant. Martin received his Master of Science degree in Spatial Information Science and Engineering in December 1997.

Since 1998 Martin has been a Graduate Research Assistant at the Institute for Geoinformation at the Vienna University of Technology, where he graduated as a Surveying Engineer (Dipl.-Ing.) in November 1998. From 1999 until now Martin has been teaching at the Institute for Geoinformation. His teaching includes courses in information systems, geometric data management, temporal data in GIS, design of GIS, assessment of GIS, and public domain GIS. He was also involved in two projects, one focusing on Multi-Agency Databases to Manage Geographic Information (COMMUTER) and the other on the establishment of a Pan European Network for Geographical Information (PANEL-GI), both funded by the European Commission. Martin has authored and co-authored several research papers published in refereed journals and conference proceedings. He was also co-editor of two books.

Martin Raubal is married and has one child. He is a candidate for the "Doktor der technischen Wissenschaften" degree from the Vienna University of Technology in 2001.